НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА СИСТЕМНОГО ПРОГРАМУВАННЯ І
СПЕЦІАЛІЗОВАНИХ КОМП'ЮТЕРНИХ СИСТЕМ

«На правах рукопису»
УДК _____

«До захисту допущено»

Завідувач кафедри СПСКС

_____ В.П.Тарасенко_
(підпис) (ініціали, прізвище)

"___"_____2018р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 123 Комп'ютерна інженерія «Системне програмування»

на тему КОМП'ЮТЕРНА СИСТЕМА ДИНАМІЧНОГО СЛІДКУВАННЯ ЗА ПОВІТРЯНИМ ТА НАДВОДНИМ ПРОСТОРАМИ

Виконала: студентка ІІ курсу, групи КВ-72мп
(шифр групи)

Клекота Олександр Олегович
(прізвище, ім'я, по батькові) (підпис)

Науковий керівник д.т.н., проф. Павловський В.І. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент_____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2018 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра системного програмування і спеціалізованих комп'ютерних систем

Рівень вищої освіти – другий (магістерський) за освітньо-професійною програмою

Спеціальність (спеціалізація) – 123 «Комп'ютерна інженерія» («Системне програмування»)

ЗАТВЕРДЖУЮ

Завідувач кафедри СПСКС

_____ В.П. Тарасенко

«___»_____2018 р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Клекоті Олександру Олеговичу

1. Тема дисертації «комп'ютерна система динамічного слідкування за повітряним та надводним просторами», науковий керівник дисертації Павловський Володимир Ілліч, д.т.н., професор, затверджені наказом по університету від «30» жовтня 2018 р. №4030-с

2. Термін подання студентом дисертації «7» грудня 2018 р.

3. Об'єкт дослідження: інформаційно-управляюча система захисту корабля від можливих загроз.

4. Предмет дослідження: алгоритм об'єднаної обробки треків цілей та алгоритми маневрування корабля.

5. Перелік завдань, які потрібно розробити:

- розгляд та аналіз існуючих систем та рішень;
- аналіз існуючих шарів базової функціональності;
- встановити задача згідно з технічними вимогами до апаратури і програмного забезпечення;
- встановити вимоги до функціональних характеристик системи;
- розробити архітектуру системи;
- спроектувати систему;
- запропонувати алгоритм об'єднання треків цілей;
- розробити комп'ютерні алгоритми маневрування кораблем;
- розробити графічний інтерфейс системи.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу:
- графічні інтерфейси системи;
- схеми взаємодії системи;
- графи зв’язків класа та графи наслідування.

7. Орієнтовний перелік публікацій:
- Тези доповіді “Комп'ютерна система динамічного відслідклвування повітряної та надводної обстановки ”
- Тези доповіді “ Комп'ютерна система динамічного відслідклвування повітряної та надводної обстановки ”

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

9. Дата видачі завдання «22» жовтня 2017 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1.	Ґрунтовне ознайомлення з предметною галуззю	25.10.2017	
2.	Визначення структури магістерської дисертації; вивчення літератури, пошук додаткової літератури, патентний пошук	17.12.2017	
3.	Робота над першим розділом магістерської дисертації; проведення наукового дослідження	04.02.2018	
4.	Проведення наукового дослідження; робота над другим розділом магістерської дисертації; розроблення програмного забезпечення	16.04.2018	
5.	Проведення наукового дослідження; робота над статтею за результатами наукового дослідження	21.05.2018	
6.	Проведення наукового дослідження; робота над третім розділом магістерської дисертації	21.06.2018	
7.	Завершення роботи над основною частиною магістерської дисертації; підготовка ілюстративного матеріалу; підготовка матеріалів доповіді на конференції ПМК-2018	09.11.2018	
8.	Оформлення текстової і графічної частини магістерської дисертації	05.12.2018	

Студент О.О. Клекота

Науковий керівник дисертації В.І. Павловський

РЕФЕРАТ

Актуальність теми Посилення обороноздатності є однією з актуальних задач України в сучасних умовах. Важливою складовою цього процесу є підсилення можливостей військово-морських сил (ВМС) по відбиттю атак можливого супротивника.

Особливістю військових кораблів є їх відкритість до нападу зі сторони надводних та повітряних сил супротивника з будь-якого напряму. Тому випереджаюче виявлення військових кораблів або літаків супротивника є життєво необхідним для успішного захисту та протидії нападу.

Виходячи з викладеного розробка корабельних систем оперативного відслідковування повітряного та надводного просторів є вкрай актуальною задачею.

В рамках програми розвитку ВМС України актуальним є створення систем динамічного відслідковування повітряного та надводного просторів кораблів типу "корвет". Основною задачею цих систем є об’єднане опрацювання надводних та повітряних цілей, надання оператору можливості легко оцінювати навколишню обстановку і допомога у вирішенні питань ефективного маневрування судном.

У зв’язку з модернізацією ВМС України встановлені жорсткі рамки частоти оновлення даних, отриманих від радіолокаторів, а саме забезпечення частоти оновлення формулярів цілей 100Гц для не менш, ніж 1000 цілей.

Об’єктом дослідження є інформаційно-управляюча система захисту корабля від можливих загроз.

Предметом дослідження є алгоритми об’єднаної обробки треків цілей та алгоритми маневрування кораблів.

Мета роботи: розробка системи динамічного відслідковування повітряного та надводного просторів та комп’ютерних алгоритмів з маневрування кораблем**.**

Методи дослідження. В роботі використовуються методи математичного моделювання, системного аналізу та чисельні методи.

Наукова новизна роботи полягає в наступному:

1. Запропоновано засоби об'єднання траєкторій цілі від різнотипних радіолокаторів, котрі забезпечують можливість точної та швидкої оцінки повітряної та надводної обстановки.

2. Запропоновані комп'ютерні алгоритми маневрування корабля, котрі забезпечують високий рівень життєздатності корабля за рахунок швидкої та точної побудови маневру.

Практична цінність. У зв'язку з потребами об'єднання треків цілей від 5 до 10 радарів був запропонований алгоритм об'єднаної обробки треків цілей, котрий задовольняє умовам точності та швидкодії, що забезпечує можливість оперативної реакції на поведінку цілі.

Реалізовані складні алгоритми маневрування, котрі допомагають командиру корабля швидко та безпомилково оцінювати можливі варіанти маневру власного судна в критичних умовах та зберігають важливий час на вирішення інших не менш критичних задач підтримки живучості корабля.

Апробація роботи. Основні положення і результати роботи були представлені та обговорювались на науковій конференції магістрантів та аспірантів та опубліковані у збірнику „Наукові вісті НТУУ «КПІ»".

Структура та обсяг роботи. Магістерська дисертація складається з вступу, шести розділів, висновків та додатків.

У вступі надано загальну характеристику роботи, виконано оцінку сучасного стану проблеми, обґрунтовано актуальність напряму досліджень.

У першому розділі проаналізована проблема відслідковування надводного та повітряного просторів та представлений аналіз існуючих систем та рішень;

У другому розділі поставлена задача згідно з технічними вимогами до апаратури та програмного забезпечення та запропоновані шляхи її вирішення.

У третьому розділі встановлені вимоги до функціональних характеристик системи динамічного відслідковування надводного та повітряного просторів.

У четвертому розділі описана архітектура та проектування системи.

У п’ятому розділі описані використані в роботі алгоритми функціонування системи.

У шостому розділі описані графічні інтерфейси системи.

У висновках проаналізовано отримані результати роботи, наведені наукова новизна та інноваційність отриманих результатів.

У додатках наведено частину коду програми.

Робота виконана на 91 аркушах, містить 3 додаток та посилання на список використаних літературних джерел з 9 найменувань. У роботі наведено 52 рисунків та 5 таблиць.

Ключові слова: алгоритм об’єднаної обробки треків, алгоритм маневрування, автоматизована система бойового управління.

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

АСБУ – автоматизована система бойового управління.

ВМС – військово морські сили.

ОТБР – основна тактично балістична ракета.

НРЛС – навігаційна радіолокаційна станція.

ПАЗ – постановник активних завад.

РЛС – радіолокаційна станція.

ТНО – тактична надводна обстановка.

ТПО – тактична повітряна обстановка.

ATHENA – architecture technologies handling electronic naval applications.

DSS – data distribution service.

CMS – command management system.

M-DLP – multi-data linc prossesor.

MFC – multi function device.

VDU – visual display unit.

DRMU – digital remote measurement unit.

CORBA – common object request broker architecture.

JMS – java management system.

QoS – quality of system.

SDK – software development kit.

ВСТУП

Посилення обороноздатності є однією з актуальних задач України в сучасних умовах. Важливою складовою цього процесу є посилення можливостей військово-морських сил (ВМС) по відбиттю атак можливого супротивника.

Особливістю військових кораблів є їх відкритість до нападу зі сторони надводних та повітряних сил супротивника з будь-якого напряму. Тому випереджаюче виявлення військових кораблів або літаків супротивника є життєво необхідним для успішного захисту та протидії нападу. Виходячи з викладеного, розробка корабельних систем оперативного відслідковування повітряного та надводного просторів є вкрай актуальною задачею.

У рамках програми розвитку ВМС України актуальним є створення систем динамічного відслідковування повітряного та надводного просторів кораблів типу «корвет». Основною задачею цих систем є третинне опрацювання надводних та повітряних цілей, надання оператору можливості легко оцінювати навколишню обстановку і допомога у вирішенні питань ефективного маневрування судном.

У зв’язку з модернізацією ВМС України встановлені жорсткі рамки частоти оновлення даних отриманих від радіолокаторів, а саме, забезпечення частоти оновлення формулярів цілей 100Гц для не менш, ніж 1000 цілей.

1. АНАЛІЗ ПРОБЛЕМ ВІДСЛІДКОВУВАННЯ ПОВІТРЯНОГО ТА НАДВОДНОГО ПРОСТОРУ КОРАБЛЯ, ІСНУЮЧИХ СИСТЕМ ТА РІШЕНЬ

1.1. Аналіз проблем відслідковування повітряного та надводного простору корабля

На сьогоднішній день існує проблема стрімкого устарівання ВМС України через малу кількість нових та глибоко модернізованих існуючих кораблів. Серед існуючих проблем можна виділити три наступні проблеми.

Через відсутність динамічної системи відслідковування повітряного та надводного простору на кораблях, в яких встановлено більше, ніж один радіолокаційний пристрій, неможливо об’єднати дані про ціль з декількох радарів, через що капітан корабля має використовувати лише один радар у конкретний момент часу, адже в іншому випадку одна ціль може бути розмножена через те, що її будуть бачити багато радарів.

Відображення даних формулярів про ціль з частотою оновлення не менше за 100Гц. Через те, що корабель у морі чи в океані уразливий для супротивника з будь-якої сторони. Вчасно побачивши загрозу, команда корабля має дуже високі шанси на протидію їй. Швидкість новітніх ракет очікуваного супротивника перевищує 1000 метрів на секунду, отже, для успішної протидії таким загрозам команда корабля має отримувати дані з затримкою не більш, як 10 мілісекунд, так як дотримання саме цього параметру дозволить зберегти життя всього екіпажу.

Відсутність комп’ютерних алгоритмів з маневрування кораблем, так як відповідальність за всі маневри корабля лягає на капітана, він власноруч має на паперовій карті розраховувати та намалювати маневри за допомогою циркуля лінійки та олівця. Через це командир корабля витрачає багато часу на розрахунки маневру, а в критичних ситуаціях цього часу завжди не вистачає. За останній рік відомо про три випадки зіткнення

військових кораблів з іншими судами. Дані події відбувалися у мирний час через неправильний розрахунок маневру, і вкрай важливо не допустити такі помилки у військових операціях.

Серед визначених проблем найбільш важливою є об’єднання даних, які надходять від різних радарів. Зокрема, на кораблях класу «корвет» зараз встановлюють від п’яти до десяти радіолокаторів різних типів, а в майбутньому кількість радарів буде збільшуватися. На кораблях встановлюються радари таких типів:

– 3D РЛС для відображення надводних і повітряних цілей;

– вузьконаправлена радіо локаційна станція (далі – РЛС) для відображення надводних і повітряних цілей;

– гідроакустичні комплекси;

– навігаційна РЛС;

– оптико-електронні системи відслідковування.

Отже, через різні похибки у різних радарах потрібно прийняти рішення чи оду й ту саму ціль бачать радари, об’єднати дані з декількох радарів, враховуючи похибки цих радарів, тобто один радар більш точно визначає координати, а інший – пеленг, потрібно створити об’єднаний формуляр цілі з найбільш точними даними.

Задача збереження даних, котрі були отримані від радіолокаторів, та дані дій операторів є досить важливою інформацією, адже після виконання місії командуванню потрібно аналізувати та оцінювати дії складу корабля і на основі цього аналізу приймати рішення про правомірність дій екіпажу в тій чи іншій ситуації. Також ці дані можливо використовувати для тренування особового складу. Слід зауважити, що за збереження даних має відповідати окремий сервер.

Через те, що система встановлюється на корабель, місії якого можуть тривати місяцями, то не менш важливою задачею виступає надійність та безвідмовність системи. Для того, щоб забезпечити високу безвідмовність

системи на кораблі, потрібно створити 2 лінії зв’язку між комп’ютерами, адже об’єднана обробка даних формулярів цілей проводиться на сервері, а відображення надводної та повітряної обстановки проводиться на комп’ютері капітана корабля та комп’ютері відображення тактичної обстановки. При умові створення двох ліній зв’язку та встановленні двох серверів виникають дві незалежні системи, котрі можуть заміняти одна одну при виході з строю однієї з систем, окрім цього маємо можливість розвантажити одну з ліній зв’язку, так як при звичайній роботі системи резервація даних буде відбуватися на резервному сервері, використовуючи резервну лінію зв’язку.

Через велике навантаження на лінію зв’язку між комп’ютерами інтерфейсом передачі прийнятий оптичний порт з протоколом ethernet.

1.2. Аналіз існуючих автоматизованих систем бойового управління

Перш за все, варто відмітити, що інформація щодо існуючих систем є дуже обмеженою. Незважаючи на це, можна зробити поверхневий аналіз існуючих автоматизованих система бойового управління (далі – АСБУ), що вже допоможе на етапі проектування системи.

1.2.1. АСБУ TACTICOS фірми Thales

TACTICOS була задумана на початку 90-х років як інтегрована і високоавтоматизована система бойового управління для освітлення тактичної обстановки навколо корабля та ефективного маневрування кораблем під час військово-морських переходів [1]. TACTICOS застосовувалася на малих, середніх і великих військових кораблях протягом трьох десятиліть з моменту її впровадження.

TACTICOS підпадає під програму постійного удосконалення для вирішення нових місій. Дана система інтегрується з різними радіолокаційними та оптоелектронними засобами.

Значною частиною цієї еволюції стала міграція на сертифіковану відкриту архітектуру OpenSplice DDS, що дозволяє TACTICOS взаємодіяти з іншими програмами і швидко приймати нові або модифіковані функціональні можливості.

Data distribution service (далі – DDS) дозволяє виконувати обмін повідомленнями в реальному часі. Цей стандарт забезпечує безперервність, своєчасність, масштабованість та стійкість системи з розподіленим обміном даними.

Thales оцінив OpenSplice DDS як midlleware з дуже високою продуктивністю в режимі реального часу, орієнтований на можливість опублікувати чи підписати проміжну платформу для виконання критично важливих дій. DDS встановлений як стандарт для передачі даних в режимі реального часу.

Поступові поліпшення дозволили зберегти TACTICOS в авангарді технологій і забезпечили TACTICOS як провідну систему на ринку АСБУ.

Широке застосування TACTICOS пояснюється великою бібліотекою функціональних можливостей і перевіреною інтеграцією обладнання.

Послужний список успіху TACTICOS охоплює понад 20 флотів і наближується до 200 платформ – від малих прибережних патрульних човнів до кораблів класу «корвет».

TACTICOS є масштабованою системою, котра задовольняє найрізноманітніші потреби користувачів. Крім того, TACTICOS має єдине ядро архітектури, котре працює на загальній апаратній платформі, призначеній для безперебійного обслуговування в режимі реального часу.

TACTICOS вирішує такі місії:

– місія Solution 100 (MS-100) для операцій з безпекою прибережної смуги;

– місія Solution 150 (MS-150) для операцій з охорони океану;

– місія Solution 300 (MS-300) для малоактивних морських операцій;

– місія Solution 400 (MS-400) для військово-морських операцій середньої інтенсивності;

– місія Solution 500 (MS-500) для інтенсивних морських операцій;

– місія Solution 1000 (MS-1000) для технічної балістичної протиракетної оборони.

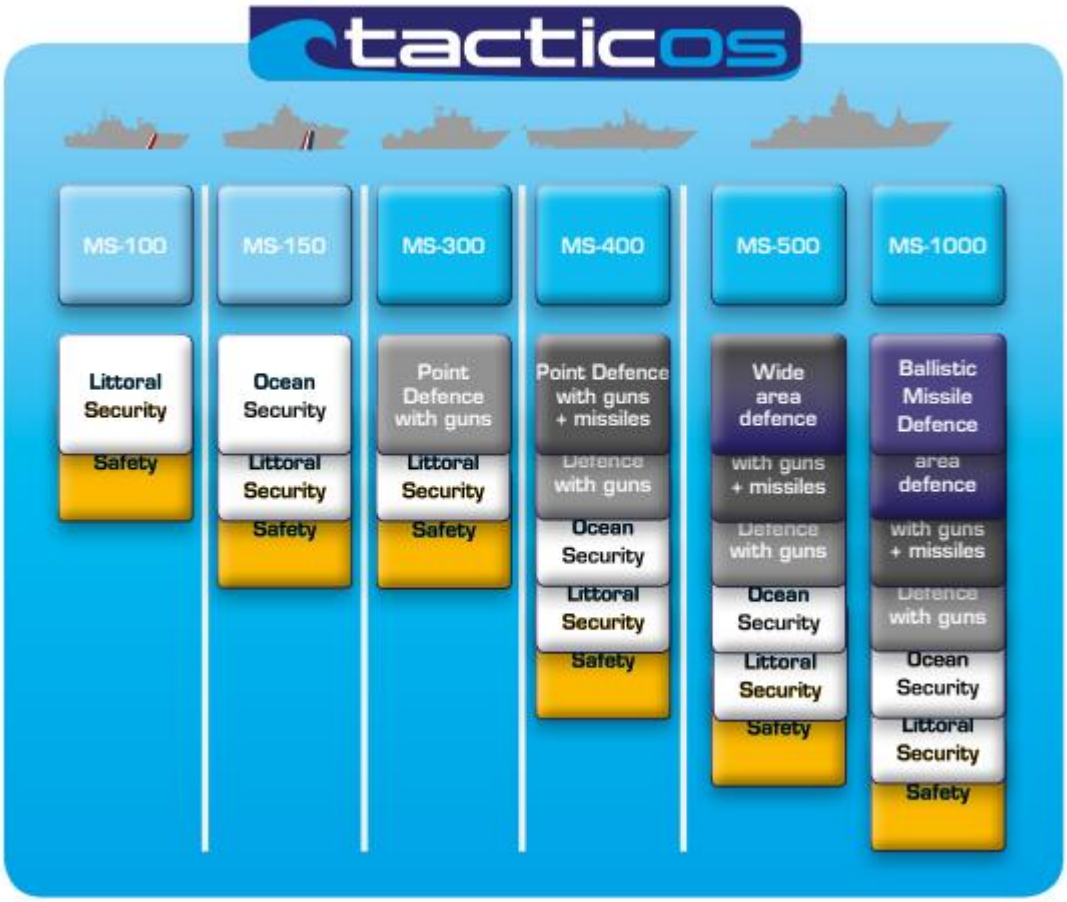

Рисунок 1.1-Класифікація місій АСБУ TACTICOS

1.2.2. АСБУ Athena фірми Leonardo

ATHENA (Architecture & Technologies Handling Electronic Naval Applications) – це масштабована АСБУ, яка використовується від патрульних суден до авіаносців. ATHENA CMS інтегрує всі функції, які є необхідними для спостереження за тактичною обстановкою, а саме: навігаційну підтримку, оцінку загроз та зброї призначення, управління зброєю, місію планування багататактних ліній передачі даних та навчання на борту [2].

Велика інтеграція військово-морських сил у міжнародному сценарії передбачає нові вимоги та обмеження про загрози, сумісність та гнучкість.

ATHENA – це сучасна АСБУ, яка призначена для задоволення всіх сучасних військових вимог та до всіх найсучасніших військово-морських програм. ATHENA CMS підходить для будь-якого класу судна за рахунок повністю резервованої модульної та масштабованої архітектурі, яка може бути налаштована відповідно до конкретної потреби.

ATHENA CMS підтримує команди управління бортовими системами та Force assets для досягнення призначених місій. ATHENA дозволяє ефективно оцінити тактичну обстановку та підтримує планування та виконання наступних операцій:

- тактичне управління інформацією;
- виконання силових обов'язків;
- управління морськими місіями в AAW, ASuW і ASW-доменах;
- управління системними ресурсами;
- системна сумісність через тактичні посилання даних та стратегічні обміни даними;
- стратегічна підтримка команд;
- навчання оператора на борту судна;
- запис та аналіз даних.

Завдяки високій продуктивності програми ATHENA охоплює всю бойову систему функцій через автоматичну інтеграцію даних, автоматичну компіляцію та оцінку тактичної обстановки разом з автоматичною координацією систем зброї через оцінку загроз та призначення зброї.

Оптимізоване управління ресурсами Combat System та широкомасштабна підтримка середньої та довгострокової стратегічної діяльності також відіграє значну роль. Система ATHENA CMS охоплює управління з інфраструктурою та послугами забезпечення продуктивності та надійності системи. Ця структура також забезпечує тактичну та стратегічну сумісність у сценаріях, орієнтованих на мережу за допомогою наступних можливостей:

– рішення TEWA допомагає підтримувати координацію військовими ресурсами, включаючи наземні сили та бойові літаки;

– компіляція спільного робочого зображення та обмін тактичною інформацією, контроль та замовлення через багатопроцесорний процесор на супутникові дані та J-REAP;

– обробка оперативних повідомлень для відображення відповідних даних;

– ATHENA CMS розкладається на функціональні сегменти, кожен з яких надає набір послуг;

– інфраструктура CMS, що забезпечує передачу і розподіл даних, відображення даних і управління пристроями введення даних, конфігурацію та моніторинг системи, розподіл відео, запис даних, відтворення та аналіз подій, генерацію та моделювання сценарію;

– зображення та спостереження за місцевою територією, управління даними та обробка даних, тактична збірка зображень на основі радіолокаційних даних, вхідні дані та посилання на дані, а також оцінка з точки зору ідентифікації та класифікації цілей;

– контроль за виконанням оперативних завдань в доменах AAW та ASuW на основі тактичної обстановки;

– tactical data link, що забезпечує ініціалізацію каналу даних, обмін даними, спостереження та командування зброєю;

– стратегічна підтримка, що забезпечує обізнаність у широкій області, допомога в плануванні місії, оперативне управління повідомленнями, інструменти співпраці.

ATHENA CMS базується на архітектурній формі, яка складається з відкритої системи, котра використовує найсучасніші апаратні та програмні технології та характеризується такими властивостями:

– високий рівень системної інтеграції та автоматизації;

– підтримка широкого кола операцій та місії;

– розгорнута інфраструктура та послуги;

– відкрита, розподілена та модульна архітектура;

– широке застосування відповідних компонентів COTS;

– додаткова тактична та планована підтримка рішень;

– інтеграція з морськими системами C4I;

– оцифровка відео (радари, телебачення).

Система приділяє велику увагу «plug and play» конструкції, використовуючи концепцію відкритої архітектури для зміни конфігурації, не впливаючи на загальну архітектуру. Обладнання ATHENA включає передачу даних по мережі, тактичні комп'ютерні блоки, мульти-процесор обробки даних (M-DLP), багатофункціональні консолі (MFC), відеорозподільчий блок (VDU), запис та управління даними (DRMU).

2. ПОСТАНОВА ЗАДАЧІ ТА ШЛЯХИ ЇЇ ВИРІШЕННЯ

2.1. Постанова задачі

Сучасні військові кораблі оснащені великою кількістю різнотипних радіолокаторів, кожен з яких відпрацьовує відповідну функцію слідкування за навколишніми надводними об'єктами. В результаті з кожного радіолокатора надходять сигнали про одні й ті ж надводні цілі. Тому з’являється необхідність у алгоритмі інтегрованої третинної обробки треків цілей, тому що кожен радіолокатор створює свій трек цілі та свою похибку, у зв’язку з чим на екрані оператора один й той же надводний об’єкт відображається як множина об’єктів, що може вводити оператора в оману.

Окрім виявлення потенційних цілей вкрай важливою є задача ефективного маневрування корабля в разі загрози атаки супротивника, що є одним з ключових факторів підтримки його живучості. При цьому основне навантаження в критичних умовах лягає на командира корабля і вимагає від нього високої кваліфікації та досвіду, що не виключає людської помилки. Саме тому актуальною є задача створення та реалізації алгоритмів маневрування задля виключення людської помилки та збереженню часу капітана корабля.

Таким чином, створення динамічної системи відслідковування повітряної та надводної обстановки вимагає вирішення наступних основних задач:

- відображення об’єктів у повітряному просторі;
- відображення об’єктів у надводному просторі;
- виконання об’єднаної обробки треків цілей;
- відображення цілі на карті формату s57;
- відображення шарів карти;
- масштабування карти;

– відображення кілець дальності;

– відображення зон ураження власної зброї;

– відображення зон освітлення обстановки власних радіолокаторів;

– відображення повного формуляру цілі;

– відображення короткого формуляру цілі;

– налаштування повного та короткого формулярів цілі;

– сигнали тривоги;

– фільтри по типу цілі;

– фільтри по ідентифікації цілі;

– поради щодо маневрування кораблем.

Окрім даних вимог також є вимоги щодо одночасного оновлення 1000 цілей та частоти оновлення формулярів цілей не нижче 100Гц. Для виконання цього завдання потрібно встановити технічні вимоги до апаратури, а саме:

– радіотехнічні засоби, котрі здатні передавати 1000 цілей;

– оптична лінія Ethernet;

– монітори з діагоналлю 32 дюйма.

Вимоги до апаратної частини комп’ютера:

– процесор Intel Xeon E5-2620 v2;

– оперативна пам’ять ddr4 ємкістю 64Гб;

– відеокарта Nvidia Quadro 4000;

– ssd накопичувачі загальним об’ємом 1Тб.

2.2. Шляхи вирішення задачі

Сучасні складні комп’ютерні системи будуються пошарово за принципом: шар прикладної функціональності (application), шар базової функціональності (middleware) та апаратура.

Найбільш критичною вимогою до динамічної системи відслідковування повітряної та надводної обстановки є оновлення даних, що надходять від радіолокаторів до комп'ютера оператора не нижче 100Гц.

У рамках проведеної розробки для забезпечення передачі даних про цілі з різних локаторів використовуються Ethernet порти. Отже, для вирішення даної проблеми можна використати один з наступних middleware сервісів, які забезпечують оновлення даних з частотою не менше 100Гц:

- sockets;
- common object request broker architecture (далі – corba);
- data distribution service (dds);
- java management system (далі – jms).

2.2.1. Сервіс поширення даних Data Distribution Service

Сервіс поширення даних **Data Distribution Service** - це стандарт передачі даних, керований OMG, що описує повідомлення з низькою затримкою для розподілених даних заявки [3].

- стандарт DDS включає підтримку типових типів даних для програми Type-Safe;
- динамічне виявлення видавців, підписчиків та тем;
- багата політика якості послуг;
- конфігурація; і на сумісність проводів.

Реалізація DDS забезпечує високу продуктивність передачі даних, є придатною для систем реального часу та в режимі реального часу. В даний час є кілька комерційних і відкритих джерел реалізації стандарту DDS, які є доступними для використання, включаючи продукти, побудовані та націлені на вбудовані комунікації.

Стандарт DDS містить простий у використанні, добре визначений інтерфейс програмування додатків. Це дозволяє розробнику написати переносний код, який буде працювати з будь-яким сумісним пристроєм. Користувачі DDS не прив'язуються до певного постачальника, а до стандарту і можуть змінювати або змішувати постачальників DDS протягом всього циклу розробки та розгортання. Загалом, DDS – це модель однорангової комунікації, у якій відсутні шлюзи та сервери, котрі повинні бути запущені або налаштовані. Історично DDS використовувався у великих системах DoD для задоволення вимог Open. Тепер із наявністю DDS для малих систем багато додатків можуть скористатися стандартизованою комунікацією publish subscribe.

2.2.2. API передачі даних Sockets

API Sockets існує протягом десятиліть і використовується практично у всіх галузях, що потребують передачі даних. Багато розробників вважають Sockets API своїм чітким вибором, котрий відповідає суворим вимогам до виконання або здійснювання зв'язку на спеціалізованих апаратних засобів та операційних системи. У вбудованих середовищах зазвичай інші комунікаційні проміжні технології не підтримуються.

2.2.3. Порівняння DDS та API Sockets

Архітектури TCP Sockets API та DDS є принципово різними.

TCP Sockets – це зв'язкова орієнтація, точка-точка, архітектура клієнт-сервер. Це вимагає, щоб клієнти та сервери знали місце розташування кожного з них для підключення, що сильно відрізняється від архітектури опублікування-підписки DDS.

З DDS розробка розподілених програм спрощується. DDS забезпечує загальне API незалежно від операційної системи або архітектури

апаратного забезпечення. DDS обробляє відкриття і управління віддаленими кінцевими точками. DDS обробляє такі деталі поведінки комунікації, як фільтрація, надійність, довговічність, збереження історії даних, виявлення неспроможності кінцевого терміну та багато іншого. Це всі функції, які не потрібно кодувати, їх налаштування доступне через стандартний API DDS, що дає більше часу, щоб зосередитися на функціональних вимогах програмного забезпечення.

Таблиця 2.1– Порівняння Socket з Data Dsitribution Service

Параметр	Socket	DDS
Архітектура	TCP: Point to point UDP: Point to point, broadcast, multicast	Publish-subscribe
Робота з різними операційними системами та мовами програмування	Для кожної операційної системи та мови програмування свій додатковий код	Одне API підтримує всі операційні системи та мови програмування
Адресація	IP addresses, port numbers жорстко закріплені	Dynamic Discovery не потрібно вказувати кінцеву точку
Безпека	Відсутня, програма має конвертувати потік байтів в правильний тип даних	Сильна безпека типів, програми викликають write() read() з певним типом даних
QoS	Спеціальний код для кожного потоку даних	Просте та повне налаштування QoS для будь-яких потоків даних

2.2.4. Технології CORBA

CORBA – це проміжне програмне забезпечення клієнта-сервера, яке базується на віддалених процедурних викликах. CORBA забезпечує шар абстракції за допомогою об'єкта брокер запитів (ORB) для керування з'єднаннями. Клієнти та сервери повинні знати один про одного безпосередньо або використовувати службу іменування.

2.2.5. Порівняння DDS та CORBA

Технології CORBA та DDS поділяють ті ж самі коріння, що і відкритий стандарт, керований системою OMG. Напевно, через це є деякі спільні риси, особливо в надійній безпеці типів даних, що забезпечується обома технологіями.

Таблиця 2.2 – Порівняння Corba з Data Dsitribution Service

Параметр	Corba	DDS
Архітектура	Point to point	Publish-subscribe
Робота з різними операційними системами та мовами програмування	Одне API підтримує всі операційні системи та мови програмування	Одне API підтримує всі операційні системи та мови програмування
Адресація	Naming Service, відсутнє Dynamic Discovery	Dynamic Discovery не потрібно вказувати кінцеву точку
Безпека	Сильна безпека типів програми викликають інтерфейси з певними типами даних	Сильна безпека типів, програми викликають write() read() з певним типом даних

Параметр	Corba	DDS
QoS	Обмежене налаштування QoS	Просте та повне налаштування QoS для будь-яких потоків даних

2.2.6. JMS служба обміну повідомленнями Java

JMS – це високорівнева структура для відправлення повідомлень між вузлами. Особливістю JMS є можливість роботи клієнта з необмеженою кількістю серверів. Для отримання даних з серверу клієнт повинен лише знати брокера та назву черги або теми.

2.2.7. Порівняння JMS та DDS

Служба обміну повідомленнями Java (JMS) та DDS є проміжним програмним забезпеченням для технології publish subscribe. Це дає певну загальну основу для порівняння двох технологій, але є також деякі істотні відмінності.

JMS використовує сервери для повідомлень, які повинні бути налаштовані з черги або теми, які будуть використовуватися. Хоча самі видавці та підписники є вільно пов'язаними, кожна заявка у комунікаціях JMS повинна підключатися до JMS-сервера. Сервер JMS обробляє відомості про підключення видавців та абонентів. Натомість DDS не вимагає жодних серверів. Використання DDS усуває потребу в сервері, а значить потенційну точку відмови і зменшує складність розгорнутої мережі.

DDS пропонує деякі значні переваги, які недоступні від пропозицій JMS, включаючи динамічне виявлення кінцевих точок; не вимагається

процес сервера; вміння адаптувати зв'язок окремих читачів даних і письменників даних через політику якості обслуговування; розширений час і фільтр на основу вмісту.

Таблиця 2.3 – Порівняння JMS з Data Dsitribution Service

Параметр	JMS	DDS
Архітектура	Publish-subscribe	Publish-subscribe
Робота з різними операційними системами та мовами програмування	Одне API підтримує всі операційні системи та мови програмування	Одне API підтримує всі операційні системи та мови програмування
Адресація	JNDI, JNS сервіси. Потрібна спеціальна конфігурація	Dynamic Discovery не потрібно вказувати кінцеву точку
Безпека	Generic Objects і XML	Сильна безпека типів, програми викликають write() read() з певним типом даних
QoS	Обмежене налаштування QoS	Просте та повне налаштування QoS для будь-яких потоків даних

При подальшому виборі middleware сервісів найважливішою характеристикою є можливість адресації типу Dynamic Discovery, яка визначає протокол розпізнавання багатоадресних програм для пошуку даних у локальній мережі, жорстка типізація даних та повне налаштування політики QoS.

На основі рекомендацій, наведених в роботі, middleware сервіс DDS (Data Distribution Service) є найкращим базовим шаром для вирішення

встановленої задачі. Додатковою його перевагою над подібними сервісами є найкраще налаштування політики QoS (Quality of service).

3. ВИМОГИ ДО ФУНКЦІОНАЛЬНИХ ХАРАКТЕРИСТИК СИСТЕМИ ДИНАМІЧНОГО ВІДСЛІДКОВУВАННЯ ЗА НАДВОДНИМ ТА ПОВІТРЯНИМ ПРОСТОРАМИ

3.1. Вимоги до архітектури та структури програми

Програма динамічного відслідковування за надводним та повітряним просторами повинна складатися з наступних складових частин:

– класи, що реалізують графічний інтерфейс користувача разом з областю відображення тактичної надводної обстановки та тактичної повітряної обстановки на фоні електронних карт, які включають в себе набір функцій для управління відображенням електронних карт тактичної надводної обстановки і тактичної повітряної обстановки, ручного додавання об'єктів і областей до складу тактичної надводної обстановки і тактичної повітряної обстановки, налаштування та виконання звукових і світлових сповіщень, курування параметрами об'єднаної обробки;

– клас, що забезпечує налаштування і відображення скорочених формулярів на фоні електронних карт;

– клас, що забезпечує налаштування і відображення повних формулярів в окремій панелі графічного інтерфейсу з можливістю редагування даних про ціль;

– класи, що забезпечують прийом / відправлення даних по мережі.

3.2. Функціональні вимоги до програми динамічного відслідковування за надводним та повітряним просторами

Програма динамічного відслідковування за надводним та повітряним просторами повинна забезпечувати вирішення наступних завдань

1. Робота з електронними картами:

– відображення електронних карт з каталогу в залежності від умов і області розміщення тактичної обстановки;

– визначення вигляду електронних карт, вибір шарів для відображення, налаштування режимів відображення в залежності від часу доби;

– масштабування, навігація по електронній карті;

– відображення поточного масштабу;

– налаштування орієнтації карти по напрямку на північ або по курсу корабля.

2. Формування та відображення тактичної надводної і повітряної обстановки:

– відображення знаків тактичної обстановки відповідно до стандартів НАТО та України;

– відображення на фоні електронних карт об'єднаної інформації про цілі;

– відображення на фоні електронних карт первинної інформації від корабельних джерел про освітлення обстановки;

– відображення траєкторій і векторів швидкості цілей;

– налаштування і відображення скорочених формулярів цілей на фоні електронних карт;

– візуальна сигналізація небезпечних цілей на фоні електронних карт;

– налаштування і відображення повних формулярів цілей з ознакою небезпеки і державною ознакою на окремій панелі графічного інтерфейсу оператора;

– перемикання між реальним і відносним режимами руху власного корабля на фоні карти;

– фільтрація цілей за класифікаційними даними, державною ознакою;

– фільтрація відображення первинної інформації від корабельних джерел освітлення обстановки по типу сенсора;

– відображення на фоні електронних карт зон ураження корабельного озброєння;

– відображення на фоні електронних карт зон видимості корабельних джерел освітлення обстановки з урахуванням гідрометеорологічної обстановки;

– відображення на фоні електронних карт шкал дальності;

– ручне додавання об'єктів і областей (полігонів) до складу тактичної надводної та повітряної обстановок;

– ручне додавання спеціальних точок і відображення даних про цілі щодо цих точок;

– ручне коректування даних про цілі;

– ручне об'єднання треків;

– відображення географічних координат поточної позиції курсора;

– відображення поточного значення шкали дальності;

– відображення мікроплану тактичної надводної та повітряної обстановок.

3. Налагодження та виконання звукової та світлової сигналізації при:

– виявленні небезпечних низьколетючих цілях;

– виявленні неопізнаної низьколетючої цілі і небезпечних плаваючих предметів;

– зміні режиму спостереження в технічних засобах і ступеня бойової готовності.

3.3. Вимоги до часових характеристик

Формування та відображення тактичної надводної обстановки має відбуватися в реальному масштабі часу. Оновлення графічного представлення тактичної надводної обстановки і тактичної повітряної обстановки має відбуватися не менше 100 разів на секунду без затримки у промальовуванні окремих елементів.

3.4. Вимоги до інформаційної та програмної сумісності системи

3.4.1. Вимоги до інформаційних структур на вході і виході

Вхідними даними для програми формування та відображення тактичної надводної обстановки і тактичної повітряної обстановки є структури (табл. 3.1), що містять всю доступну інформацію про об'єкти тактичної обстановки, що надходить від сенсорів і пройшли через процес об'єднання для уточнення даних про об'єкт, та їх відображення і відображення самих об'єктів на фоні електронних карт.

Для відображення тактичної надводної обстановки і тактичної повітряної обстановки в якості фону використовуються векторні електронні навігаційні карти (ENC) в форматі S-57 [4]. Основне призначення стандарту S-57 – стандартизація обміну гідрографічними даними між гідрографічними службами, агентствами, виробниками картографічної продукції та ECDIS-системами. Будучи стандартом обміну гідрографічними даними, S-57 є не оптимальним при прямому

використанні в суднових навігаційних системах. Навігаційні електронно-картографічні системи можуть використовувати внутрішній формат представлення даних – SENC (System ENC). Формат SENC є більш компактним і спеціально призначений для подання картографічної інформації на екрані монітора. У нашому випадку S-57 з сумісним SENC-форматом є базовим для інструментарію розробки ГІС-додатків ГІС Конструктор формат картографічних даних SIT [5].

Електронна карта може складатися з набору окремих аркушів (планшетів). Вміщені в одну базу даних листи цифрової карти утворюють район робіт. Листи карти одного району робіт повинні бути одного масштабу, проекції, системи координат.

Інструментарій для розробки ГІС додатків має вбудований функціонал для конвертації електронних карт з формату S57 у формат SIT.

Модуль повинен отримувати список неопізнаних низьколетючих цілей і небезпечних плаваючих предметів для виконання світлового і звукового оповіщення в графічному інтерфейсі.

Модуль повинен отримувати структуру даних з ознаками про зміну і поточними режимами спостереження в технічних засобах та ступенями бойової готовності.

Загальна структура формуляра цілі представлена в таблиці 3.1.

Таблиця 3.1 – Структура об'єднаного формуляру цілі

№	Параметри повідомлення	Ім’я поля структури	Тип даних	Довжина
1	Номер цілі в загально корабельній нумерації	number	unsigned long	32
2	Час формування даних про ціль	packageTimeStamp	unsigned long	64
3	Дальність	distance	unsigned long	32
4	Азимут цілі	bearingAngle	float	32
5	Кут місця цілі	elevationAngle	float	32
6	Координата цілі X	x	long	32
7	Координата цілі Y	y	long	32
8	Координата цілі Z	z	long	32
9	Вектор швидкості цілі Vx	vx	float	32

№	Параметри повідомлення	Ім’я поля структури	Тип даних	Довжина
10	Вектор швидкості цілі Vy	vy	float	32
11	Вектор швидкості цілі Vz	vz	float	32
12	Тип цілі	type	octet	8
13	Підтип цілі	subtype	octet	8
14	Рівень безпеки цілі	threatLevel	octet	8
15	Параметри ДО цілі	affiliation	octet	8
16	Признак маневру цілі	isManeuvering	boolean	8
17	Признак відміни знищення цілі	cancelDestroyingFlag	octet	8
18	Признак назначення на знищення цілі	destroyingFlag	octet	8
19	Стан треку	trackState	octet	8
20	Тип треку цілі	trackType	octet	8
21	Признак імітованої цілі	simulationFlag	octet	8
22	Признак цілі що імітує	fakeTargetFlag	octet	8
23	Час до максимального зближення з ціллю	tcpa	unsigned long	32
24	Курсовий параметр цілі	cpa	unsigned long	32
25	Курс цілі	course	float	32
Список треків сенсорів, використаних при коріляції треку (0…N - 1)				
…	……		……	...
…	Ідентифікатор платформи сенсора	platformId	octet	8
…	Ідентифікатор сенсора	sensorId	octet	8
…	Номер траекторії в нумерації сенсору	sensorBasedNumber	unsigned short	16
…	……		……	…

Інформаційними структурами на виході повинні бути формуляри об'єктів тактичної надводної обстановки і тактичної повітряної обстановки з внесеними оператором коригуваннями даних про цілі, що керують пакетами для об'єднаної обробки.

4. АРХІТЕКТУРА ТА ПРОЕКТУВАННЯ СИСТЕМИ

Архітектура системи складається з таких основних складових:

– апаратна система;

– програмна система.

4.1. Схема взаємодії апаратної системи

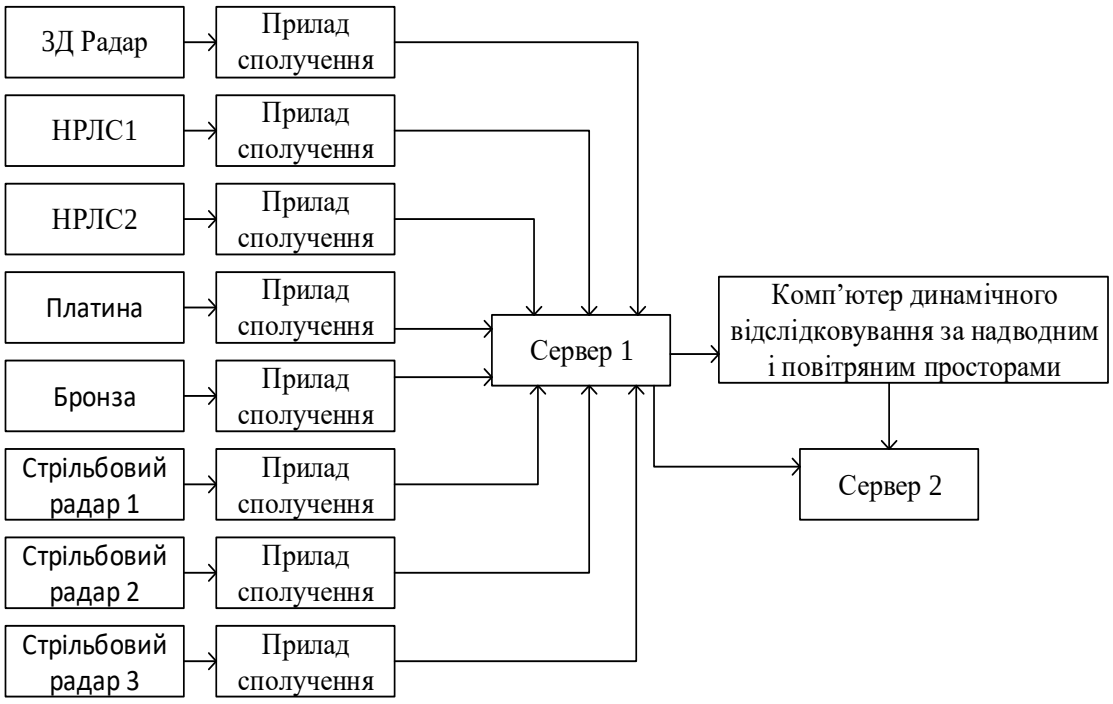

Рисунок 4.1 – Блок схема взаємодії приладів

Прилад сполучення – це плата котра потрібна для зміни інтерфейсу видачі інформації з інтерфейсу радару на оптичний інтерфейс ethernet.

Сервер 1 – це сервер на котрому відбувається об'єднана обробка треків цілей.

Сервер 2 – це сервер збереження інформації по ціль та про дії оператору.

4.2. Схема взаємодії програмної системи

Рисунок 4.2 – Блок схема програмної системи

– Апаратні засоби – це радіолокаційні пристрої котрі забезпечують систему вхідною інформацією;

– Midleware service було обрано DDS, він виконує маршрутизацію та передачу даних від апаратних засобів до комп’ютеру динамічного відслідковування за надводним та повітряним простором та серверів;

– Програма динамічного відслідковування за надодним і повітряним просторами виконує налаштування фільтрів та даних длдя відображення;

– Відображення – відображення виконується на сучасних моніторах з матрицями типу IPS.

4.3. Організація вхідних і вихідних даних

Вхідними даними для формування і відображення тактичної надводної обстановки і тактичної повітряної обстановки є:

– формуляри надводних цілей від локальних сенсорів (безпосередньо з сенсорами сполучаються модулі обробки даних від сенсорів, які взаємодіють з ними відповідно до протоколів сполучення, після чого ця інформація перетворюється в структуру даних певного виду і передається по мережі з використанням засобів OpenSplice DDS).

1. Джерелами даних є наступні сенсори:
   – виріб «Фенікс-У»;
   – виріб «Позитив-У1»;
   – система «протазани-К»;
   – система «Селена»;
   – НРЛС;
   – комплекс «Морена»;
   – система «Спис-К».
2. формуляри цілей після об'єднання і вироблення рекомендацій щодо оцінки тактичної надводної обстановки і тактичної повітряної обстановки;
3. дані з файлів каталогу електронних карт;
4. список цілей від модуля видачі звукового та світлового сигналів оповіщення при виявленні неопізнаної повітряної цілі і небезпечних плаваючих предметів;
5. дані про зміну режиму спостереження в технічних засобах і ступеня бойової готовності;
6. структури даних з тактичними розрахунками дальності виявлення цілей з урахуванням гідрометеорологічної обстановки радіолокаційної системи освітлення.

Вихідні дані з урахуванням ручного коректування даних про ціль у вигляді скоригованого формуляра цілі передаються в модуль об'єднаної обробки інформації засобами OpenSplice DDS.

Також як вихідні дані виступає керуючий пакет для модуля об'єднаної обробки інформації.

4.4. Вибір операційної системи

Системи динамічного відслідковування за надводним і повітряним просторами можуть базуватися на наступних операційних системах:

- Windows;
- Linux.

Через вимоги безпеки до програм, котрі розробляються для ВМС України, весь код проекту та всі використані бібліотеки мають бути відкриті. Внаслідок цього не можливо використовувати операційну систему Windows через її закриті бібліотеки.

Linux був обраний як операційна система, котра задовольняє вимоги ВМС України. Існує багато різних реалізацій операційних систем Linux:

- Red Hat Enterprise Linux;
- CentOs;
- Fedora;
- Mint;
- Debin;
- Ubuntu;
- QNX.

Зазвичай задля найкращої швидкодії та досягнення рамок системи реального часу використовують QNX. Проте, через вибір midlleware DDS неможливо використовувати QNX. Серед інших операційних систем Red Hat Enterprise Linux зарекомендував себе як система з найкращим рівнем підтримки. Основною перевагою над конкурентами стала підтримка системи 24/7. Отже, як операційну систему для проекту було обрано Red Hat Enterprise Linux Workstation Release 7.X.

4.5. Вибір мови програмування

Вкрай важливо обрати мову програмування, котра буде задовольняти потрібний рівень швидкодії. Зазвичай використовують такі мови програмування:

– C++;
– C;
– C#;
– Java.

4.5.1. Порівняння мов C та C++

На мові C++ можливо запускати більшу частину коду C, проте це неможливо зробити у зворотному напрямі. C++ підтримує процедурне і об’єктно орієнтоване програмування, в той час як мова C підтримує лише процедурне. C – це функціонально керована мова, а C++ є об’єктно керованою. C++ дані та функції інкапсулюються у формі об'єкта, а клас є орієнтиром об'єкта. В C дані та функції є окремими та вільними сутностями [6]. На відміну від C, у C++ неможливо застосовувати багаторазове об’явлення глобальних змінних, доступна перегрузка функцій та операторів, функції можуть бути описані в структурі. В C++ присутня зміна namespace за для можливості використання однаково названих змінних у різних частинах проекту, а також підтримуються Reference перемінні та Virtual функції.

4.5.2. Порівняння мов C++ та C#

Розмір бінарних файлів. C# має багато накладних витрат та бібліотек, до яких вона буде звертатись ще до компіляції. C ++ є набагато

легшою. Тому після того, як вона компілює у порівнянні з C ++, Binary файли C# є значно більшими за розміром [7].

Продуктивність: C ++ широко використовується, в той час як мови вищого рівня є не ефективними. C ++ код має кращу швидкодію, ніж код C #, що робить його кращим рішенням для програм, де важлива продуктивність.

Виділення пам’яті. В C ++ потрібно виділяти пам'ять для своїх об'єктів та потім звільняти її, в C# пам’ять виділяється та звільняється автоматично.

Мета платформи: програми C#, як правило, орієнтовані на операційну систему Windows, хоча Microsoft працює над крос-платформою підтримки програм C#. За допомогою C ++ ви можете кодувати для будь-якої платформи, включаючи Mac, Windows і Linux.

Типи проектів: програмісти C ++ зазвичай зосереджують увагу на програмах, що працюють безпосередньо з апаратним забезпеченням або потребують кращої продуктивності, ніж інші мови можуть запропонувати. Програми C ++ включають додатки на сервері, мережі, ігрові пристрої та навіть драйвери пристроїв для ПК . C# зазвичай використовується для веб, мобільних та настільних додатків.

4.5.3. Порівняння мов C++ та Java

Java – це дійсно справжня та повна об'єктно-орієнтована мова; в той час як C ++ є розширенням C з об'єктно-орієнтованою поведінкою. Відмітимо, що глобальні змінні можна оголосити в C ++, в Java – ні.

Деструктор об'єкта: Java руйнує об'єкт у методі finalize, тоді як C + + руйнує об'єкт через деструктори. Файли C++ поділяються на файли заголовків та файли коду програми, в Java відсутні файли заголовки [9].

Структури та об'єднання (structures and unions): C ++ підтримує структури та об'єднання, Java – ні [8]. Перевантаження оператора

підтримується в C ++ і не підтримується в Java. C ++ підтримує умовну компіляцію, Java – ні. Java має вбудовану підтримку потоків, але C ++ не має такої вбудованої підтримки. C ++ підтримує аргументи за замовчуванням, Java – ні. C ++ має оператора роздільної здатності (: :), Java – ні. C ++ має оператора переходу goto, Java – ні.

На основі порівняння запропонованих мов програмування мовою для проекту був обраний C++(ISO/IEC 14882:1998).

4.6. Вибір середовища розробки

Існує два середовища розробки для мови C++ для операційної системи Linux:

– Qt Creator;

– Eclipce.

У кожного IDE є свої переваги, проте, ключовою перевагою став редактор форм, який присутній в Qt Creator, завдяки якому можна легко та швидко створювати графічні інтерфейси програми. Також ще однією перевагою є наявність слотів та сигналів (slot, signal). Середою розробки програми обрано Qt SDK version 1.1.4.

4.7. Сторонні бібліотеки

Для відображення надводної та повітряної обстановки на фоні карти потрібно використовувати сторонні бібліотеки. Як додаткову бібліотеку для відображення векторної карти формату s57 було обрано бібліотеку ГИС toolkit для Qt Designer version 10.7.5.

4.8. Опис логічної структури

4.8.1. Алгоритм програми динамічного відслідковування за надводним і повітряним просторами

При запуску програми виконується завантаження карти в форматі s57 і приймаються налаштування за замовчуванням для інтерфейсу даного модуля і модулів, пов'язаних з даними.

Відображення електронної карти разом з тактичною надводною обстановкою (далі – ТНО) і тактичною повітряною обстановкою (далі – ТПО) відбувається з використанням подвійної буферизації. Механізм роботи подвійної буферизації наступний - при завантаженні електронної карти виконується промальовування її частини, розміром із область відображення в графічному інтерфейсі користувача і з заданою початковою точкою, у 2 буфери в оперативній пам'яті. Промальовування об'єктів виконується в другому буфері, після чого зображення з другого буфера відображається у вікні карти графічного інтерфейсу користувача. При зміні ситуації з одними об'єктами виконується затирання зображення в другому буфері зображенням з першого, промальовування нових об'єктів у другому буфері і відображення зображення з другого буфера у вікні карти графічного інтерфейсу користувача.

В ході роботи програми, циклічно з частотою 10 Гц, зі сховища даних в оперативній пам'яті виконується зчитування актуальних формулярів цілей, даних про наш корабель і даних про рівень небезпеки і класифікації цілей. На базі цих даних створюються графічні уявлення об'єктів. Далі, з використанням механізму подвійної буферизації, виконується їх промальовування і (в залежності від налаштувань, виконаних користувачем через графічний інтерфейс) промальовування векторів швидкості для об'єктів, зон освітлення, зон ураження, шкали дальності.

У панелі в нижній частині графічного інтерфейсу виводиться список ранжируваних цілей з можливістю їх автоматичного і ручного сортування.

При натисненні лівої кнопки мишки по цілі з'являється (налаштовується через меню графічного інтерфейсу) пов'язаний з ціллю скорочений формуляр. При натисканні правої клавіші мишки по цілі в панелі в правій частині графічного інтерфейсу з'являється повний формуляр цілі. Через повний формуляр цілі проводиться його редагування в залежності від прав оператора. Коректури для формуляра передаються в сховища даних в оперативній пам'яті для їх подальшого використання модулем об'єднаної обробки.

4.8.2. Структура програми

Класи, які були використані в програмі динамічного відображення надводної та повітряної обстановки:

– FIFOWrapper;
– FormularWidget;
– QDMapViewPort;
– QDMapViewWindow;
– RankingBar;
– ReadFromFIFOThread;
– RedisSortedSet< T >;
– SemWrapper;
– SideBar;
– TrackHandler< T >.

Нижче наведені основні класи діаграми програми динамічного відображення надводної та повітряної обстановки:

– клас FIFOWrapper (рис. 4.3);

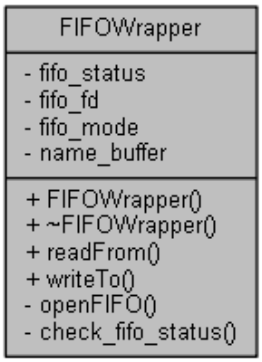

Рисунок 4.3 – Граф зв'язків класу

– клас FormularWidget(рис. 4.4);

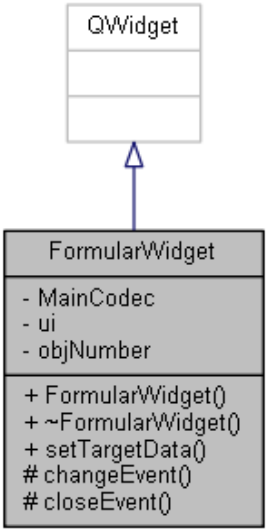

Рисунок 4.4 – Граф зв'язків класа

– клас QDMapViewPort (рис. 4.5 і 4.6);

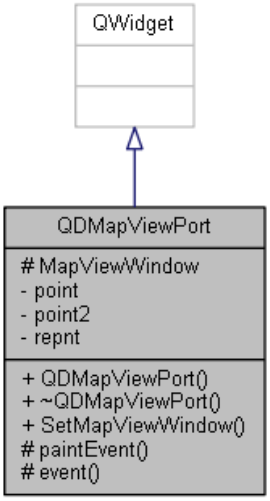

Рисунок 4.5 – Граф наслідування

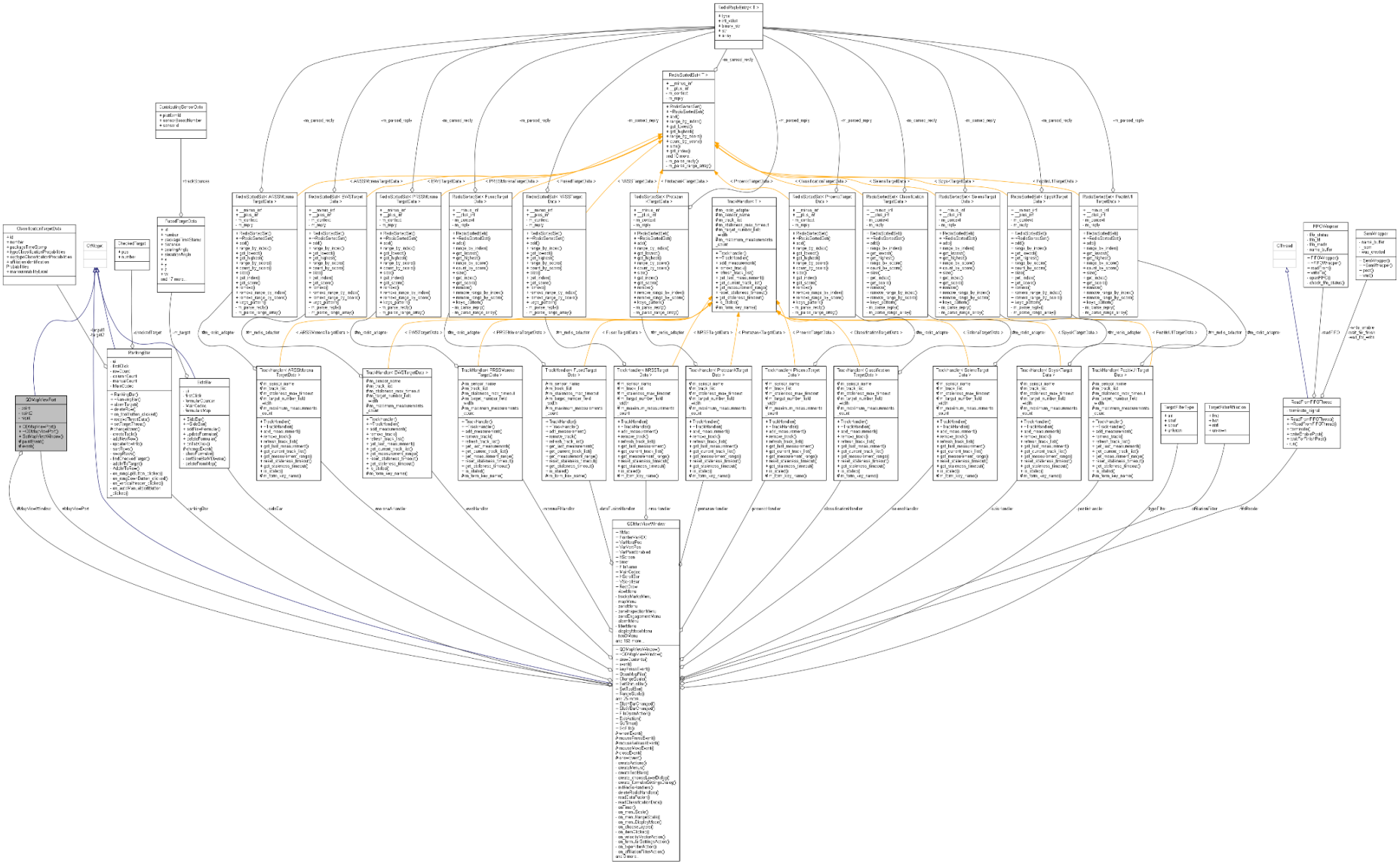

Рисунок 4.6 – Граф зв’язків класа

– клас QDMapViewWindow (рис. 4.6);

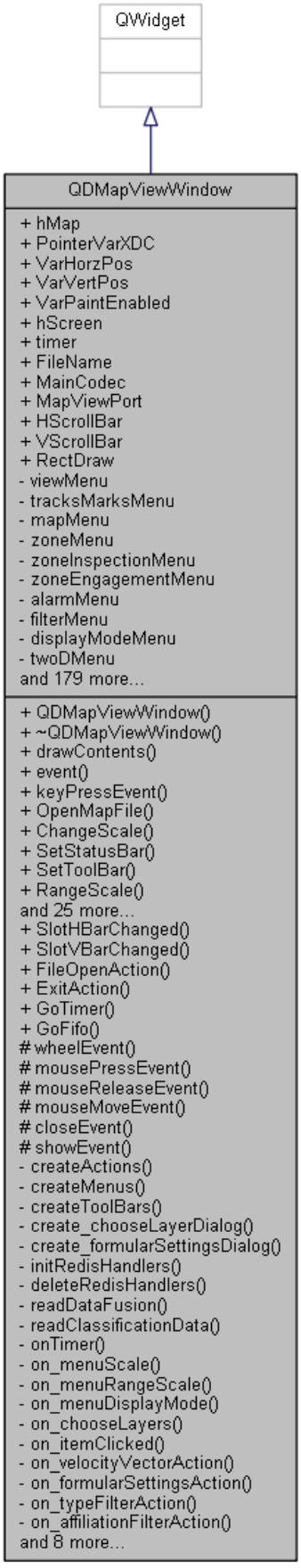

Рисунок 4.7 – Граф наслідування

– клас RankingBar (рис. 4.8 і 4.9);

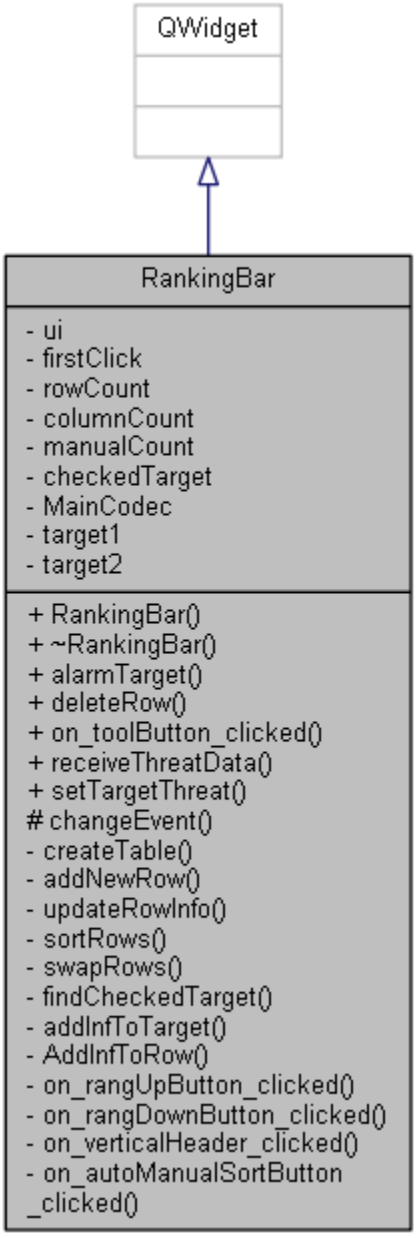

Рисунок 4.8 – Граф наслідування

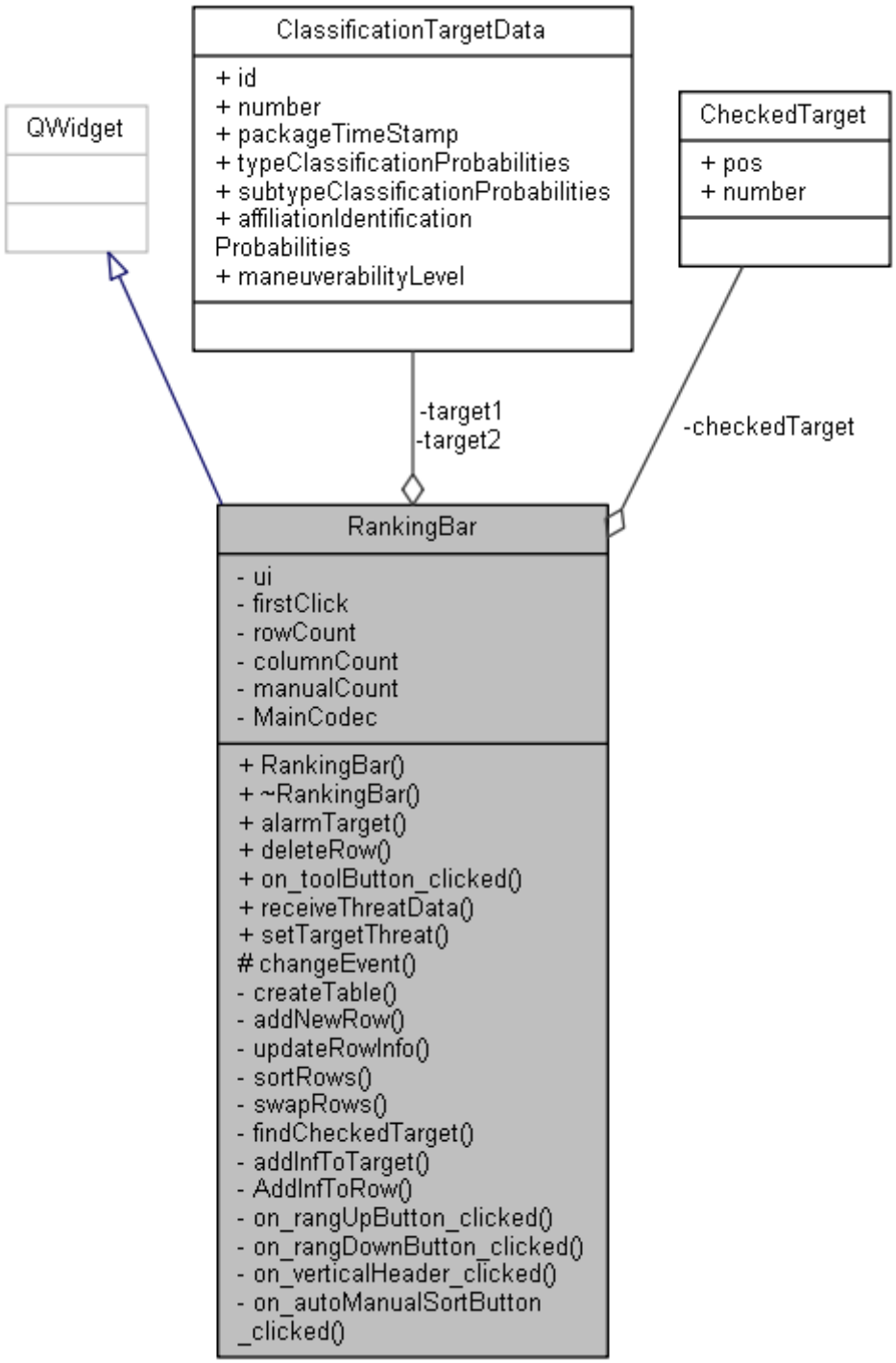

Рисунок 4.9 – Граф зв'язків класа

– клас ReadFromFIFOThread (рис. 4.10 і 4.11);

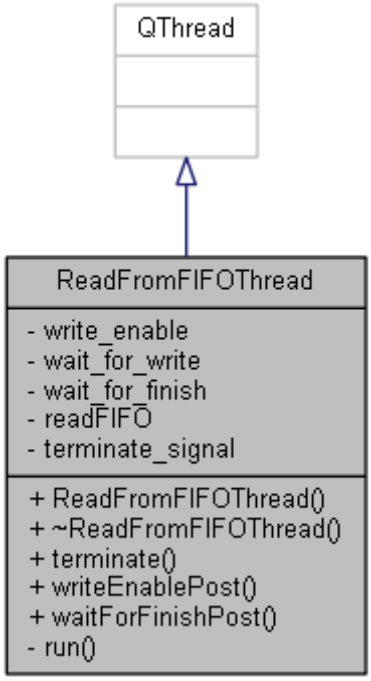

Рисунок 4.10 – Граф наслідування

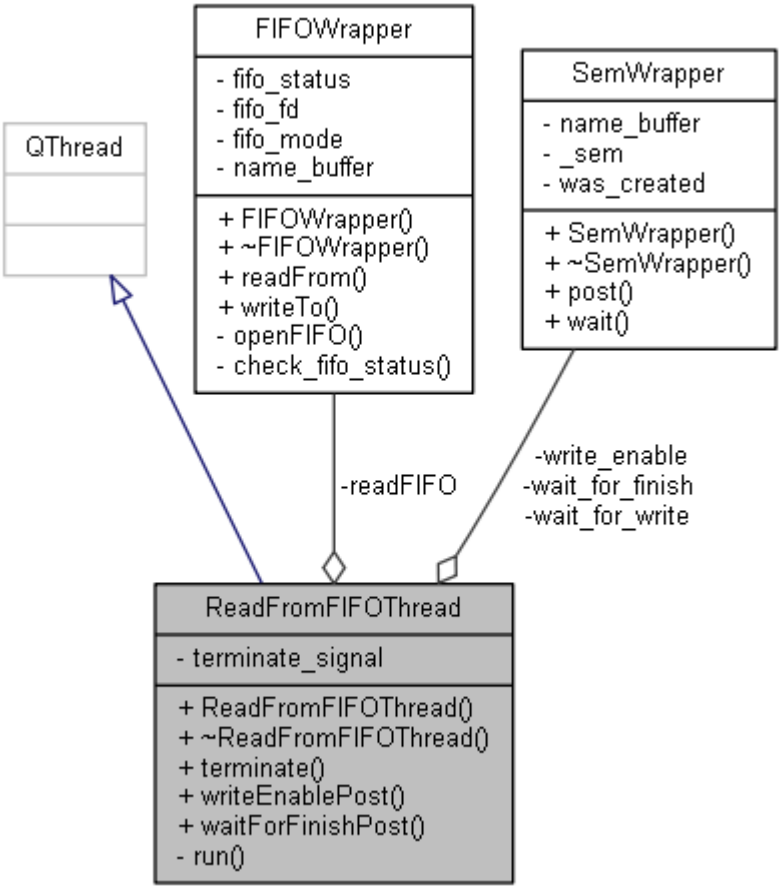

Рисунок 4.11 – Граф зв'язків класа

– клас RedisSortedSet< T > (рис. 4.12 і 4.13);

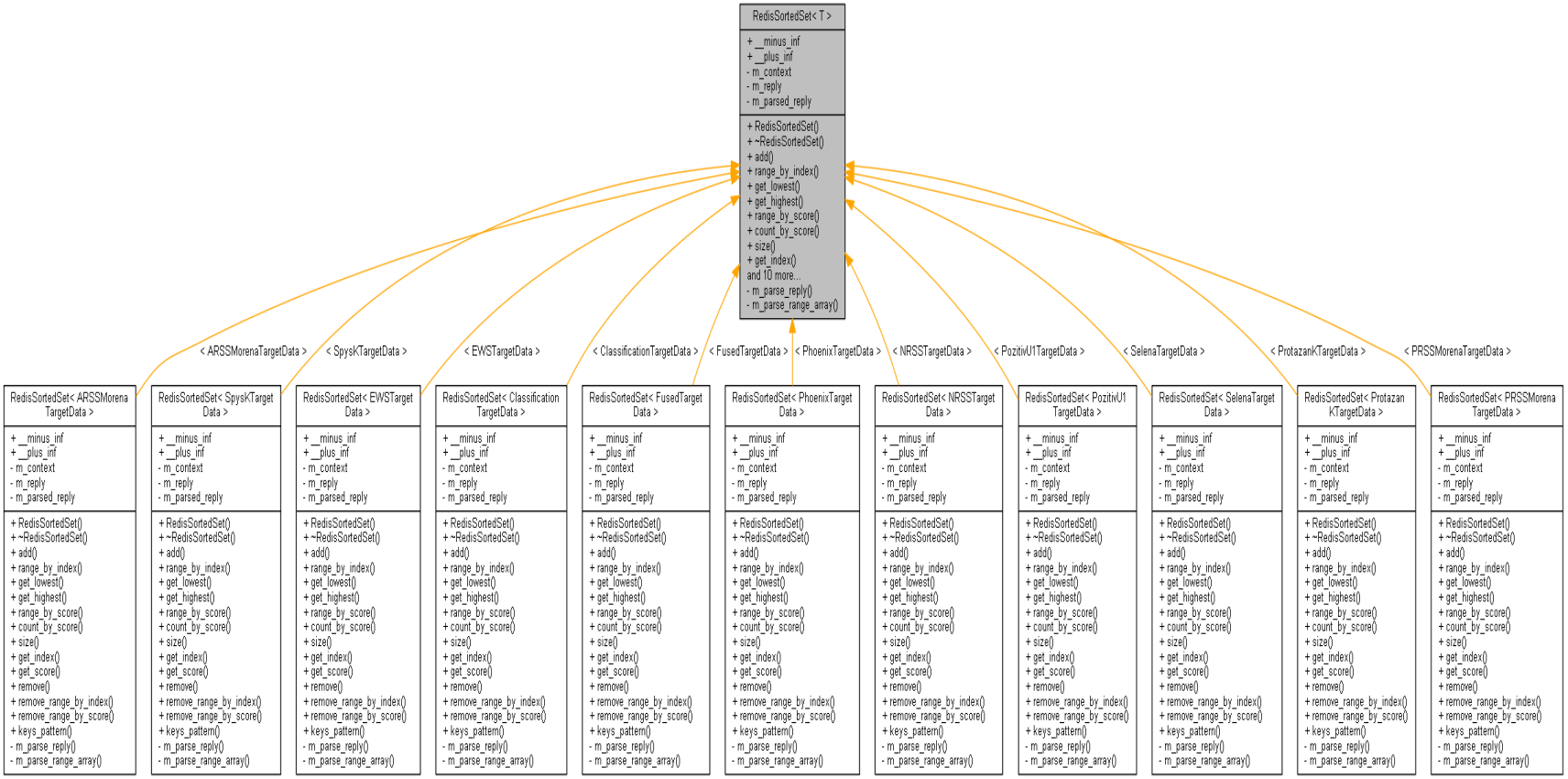

Рисунок 4.12 – Граф наслідування

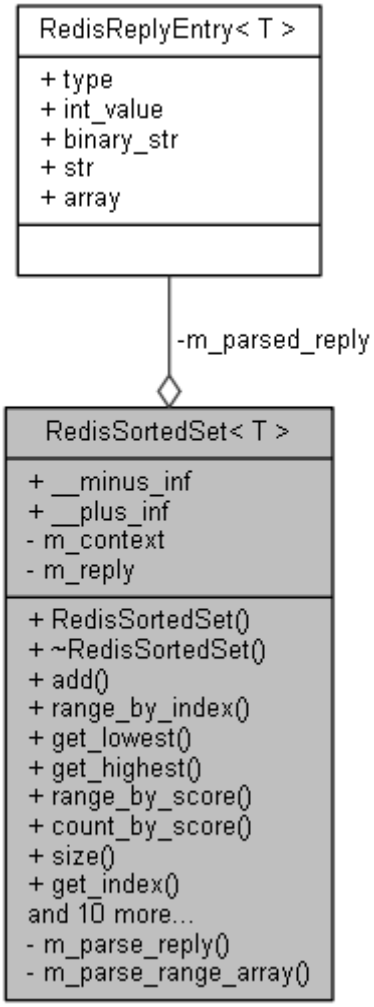

Рисунок 4.13 – Граф зв'язків класа

– клас SemWrapper (рис. 4.14);

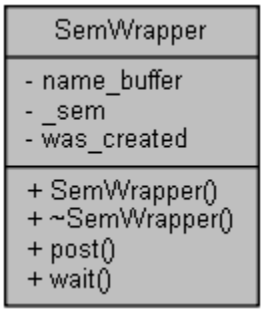

Рисунок 4.14 – Граф зв'язків класа

– клас SideBar (рис. 4.15 і 4.16);

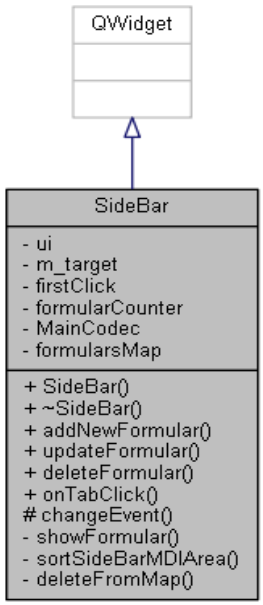

Рисунок 4.15 – Граф наслідування

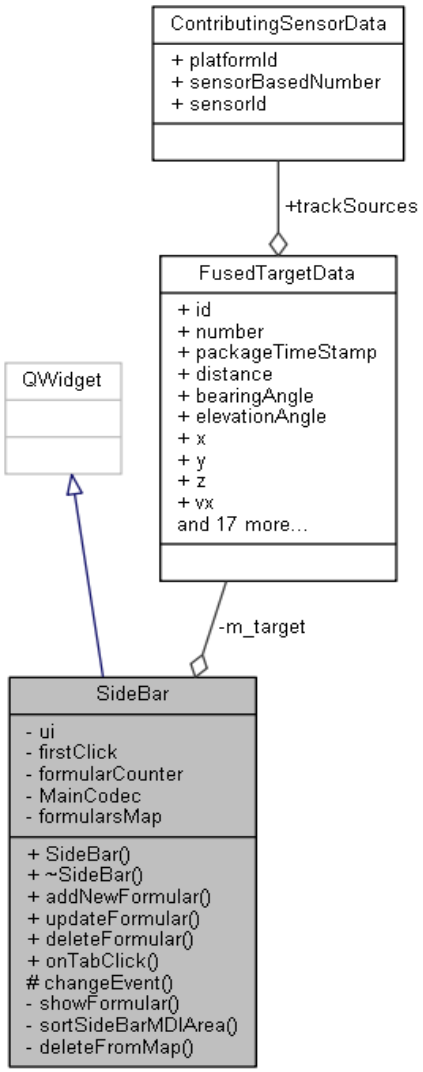

Рисунок 4.16 – Граф зв'язків класа

– клас TrackHandler< T > (рис. 4.17 і 4.18).

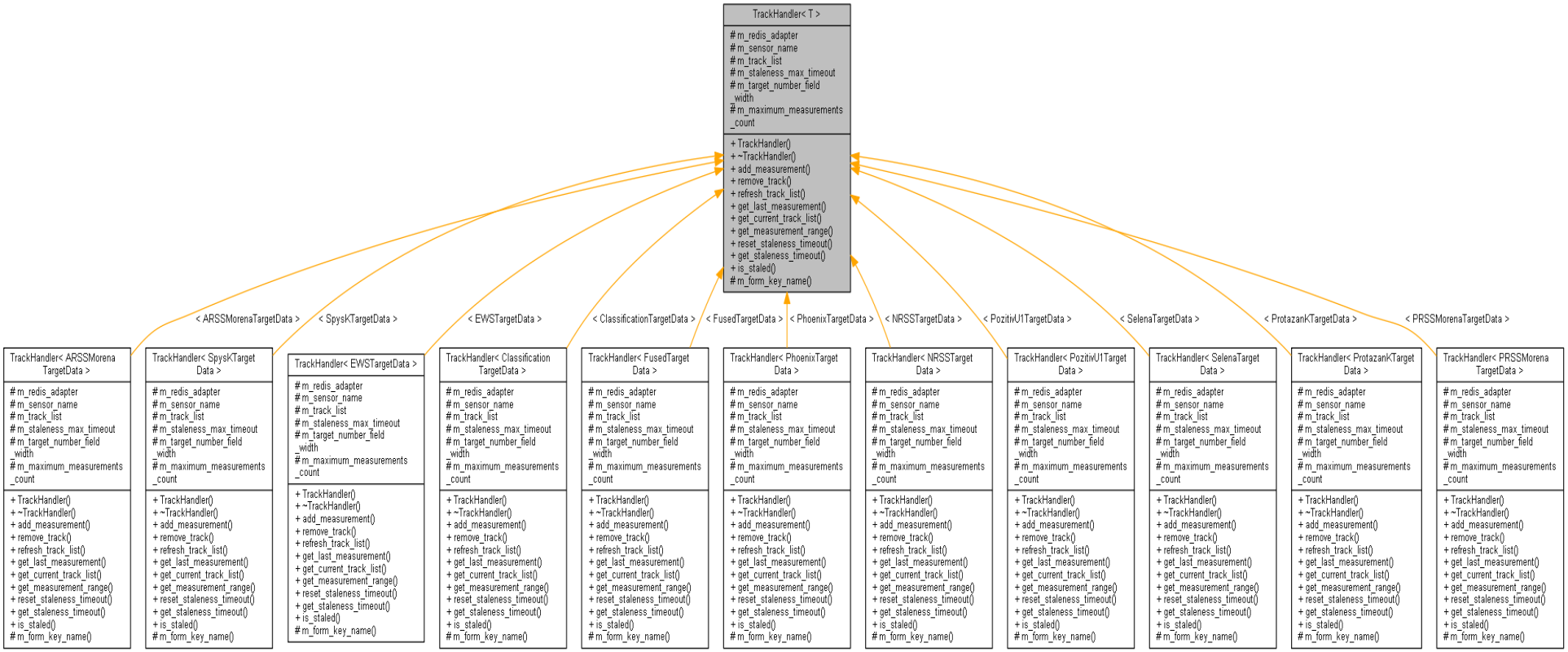

Рисунок 4.17 – Граф наслідування

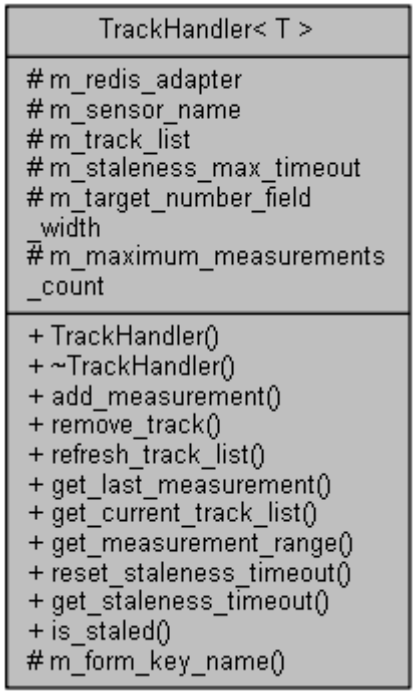

Рисунок 4.18 – Граф зв'язків класа

Основні дії з початкової ініціалізації і подальшого запуску робочих функцій модуля виконуються в конструкторі класу головного вікна програми QDMapViewWindow. На малюнку 4.19 зображений граф викликів функцій для конструктора класу QDMapViewWindow.

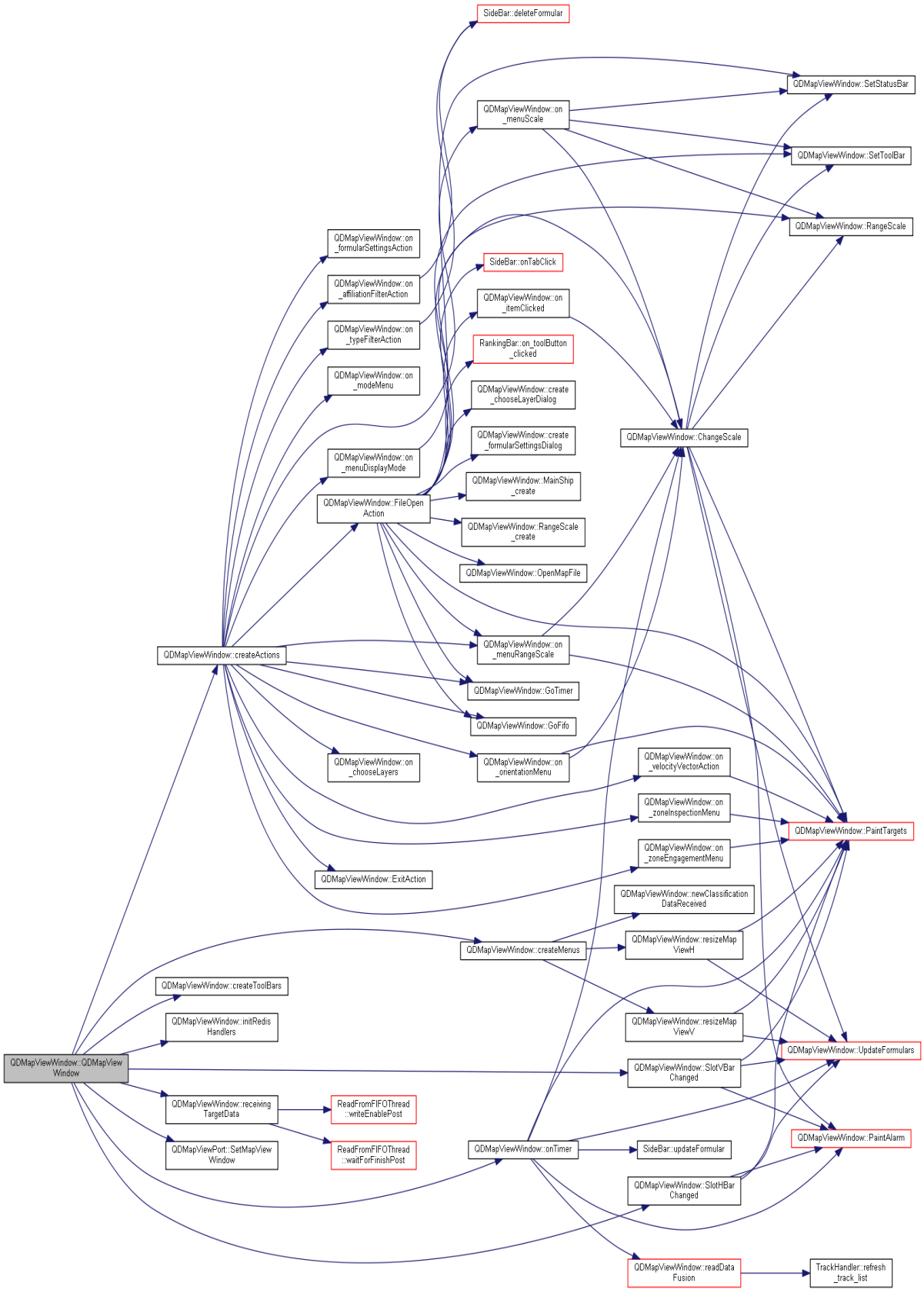

Рисунок 4.19 – Граф викликів функцій для конструктора

void ChangeScale(float Change) – функція призначена для кратної зміни масштабу відображення електронної карти.

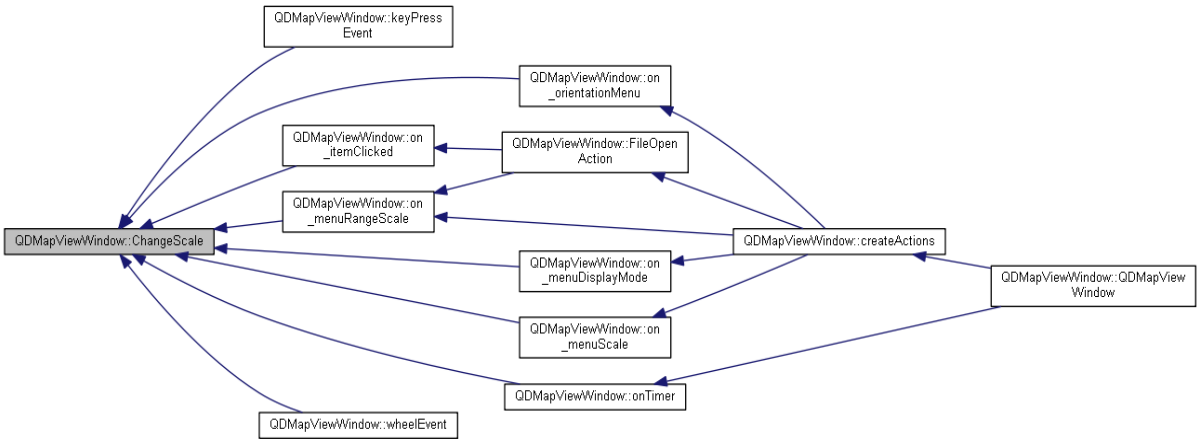

Рисунок 4.20 – Граф викликів функцій

void create_chooseLayerDialog() – функція призначена для створення діалогового вікна для вибору відображуваних шарів карти.

Рисунок 4.21 Граф виклику функції ChangeScale(float Change)

void create_formularSettingsDialog() – функція призначена для створення діалогового вікна налаштування скороченого формуляра.

Рисунок 4.22 – Граф виклику функції create_formularSettingsDialog()

void CreateDeleteFormular(int it, double cx, double cy, QMap<int, QTableWidget*> *fm, QVector<FusedTargetData> *trg) – функція призначена для створення, відображення скороченого формуляра цілі на фоні електронної карти і його знищення.

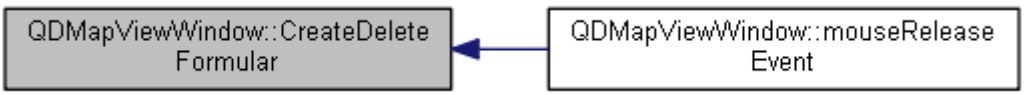

Рисунок 4.23 – Граф виклику функції CreateDeleteFormular(int it, double cx, double cy, QMap<int, QTableWidget*> *fm, QVector<FusedTargetData> *trg)

void drawContents(QPainter* p, int cx, int cy, int cw, int ch) – функція призначена для промальовування зображення карти і призначених для користувача об'єктів з буфера у вікні відображення електронної карти головного вікна програми.

Рисунок 4.23 – Граф виклику функції drawContents(QPainter* p, int cx, int cy, int cw, int ch)

void QDMapViewWindow::FileOpenAction() – функція виконує завантаження електронної карти з файлу.

Рисунок 4.24 – Граф виклику функції QDMapViewWindow::FileOpenAction()

void QDMapViewWindow::initRedisHandlers() – функція виконує ініціалізацію об'єктів, необхідних для взаємодії зі сховищем в оперативній пам'яті.

Рисунок 4.25 – Граф виклику функції QDMapViewWindow::initRedisHandlers()

void MainShip_create() – функція призначена для створення графічного представлення власного корабля для промальовування на фоні карти.

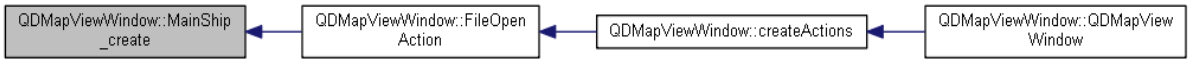

Рисунок 4.26 – Граф виклику функції MainShip_create()

void on_affiliationFilterAction() – функція призначена для фільтрації відображуваних цілей по держопізнанню.

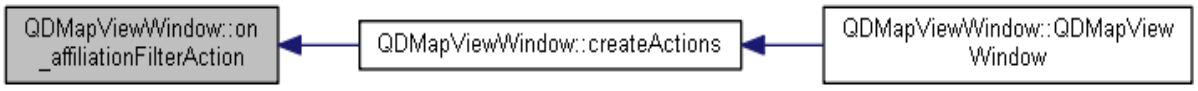

Рисунок 4.27 – Граф виклику функції on_affiliationFilterAction()

void on_menuDisplayMode() – функція призначена для зміни режиму відображення електронної карти.

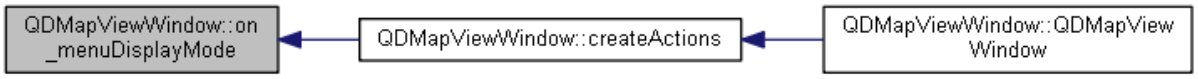

Рисунок 4.28 – Граф виклику функції on_menuDisplayMode()

void on_menuRangeScale() – функція призначена для ручної зміни масштабу шкали дальності.

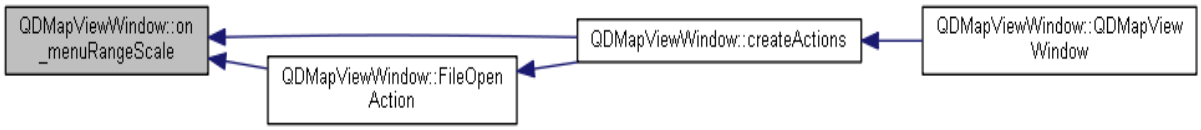

Рисунок 4.28 – Граф виклику функції on_menuRangeScale()

void on_modeMenu() – функція призначена для зміни режиму руху власного корабля.

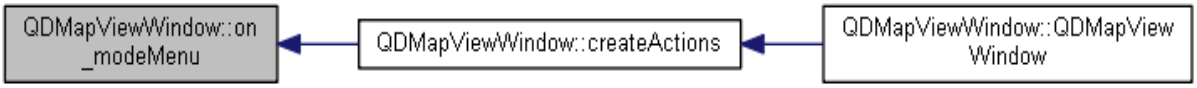

Рисунок 4.29 – Граф виклику функції on_modeMenu()

void on_orientationMenu() – функція призначена для зміни режиму орієнтації карти.

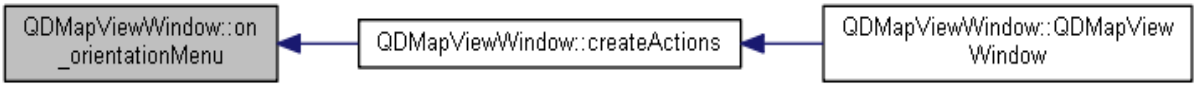

Рисунок 4.30 – Граф виклику функції on_orientationMenu()

void on_typeFilterAction() – функція призначена для фільтрації відображуваних цілей по типу.

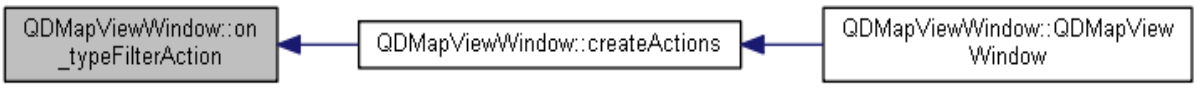

Рисунок 4.31 – Граф виклику функції on_typeFilterAction()

void on_velocityVectorAction() – функція призначена для відображення на фоні карт векторів швидкості цілей.

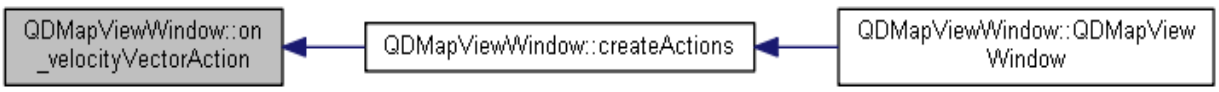

Рисунок 4.32 – Граф виклику функції on_velocityVectorAction()

void PaintAlarm() – функція призначена для виконання світлової та звукової сигналізації в інтерфейсі програми.

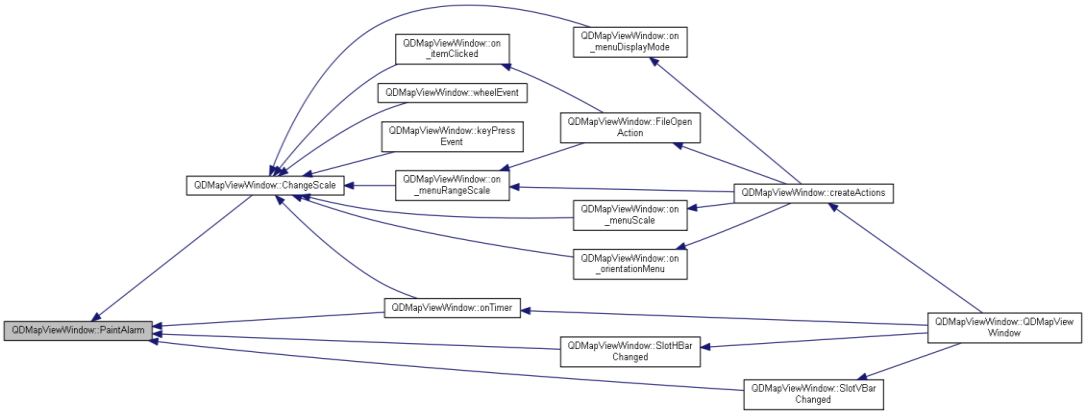

Рисунок 4.33 – Граф виклику функції PaintAlarm()

void PaintBearingTargets() – функція призначена для промальовування пеленгів.

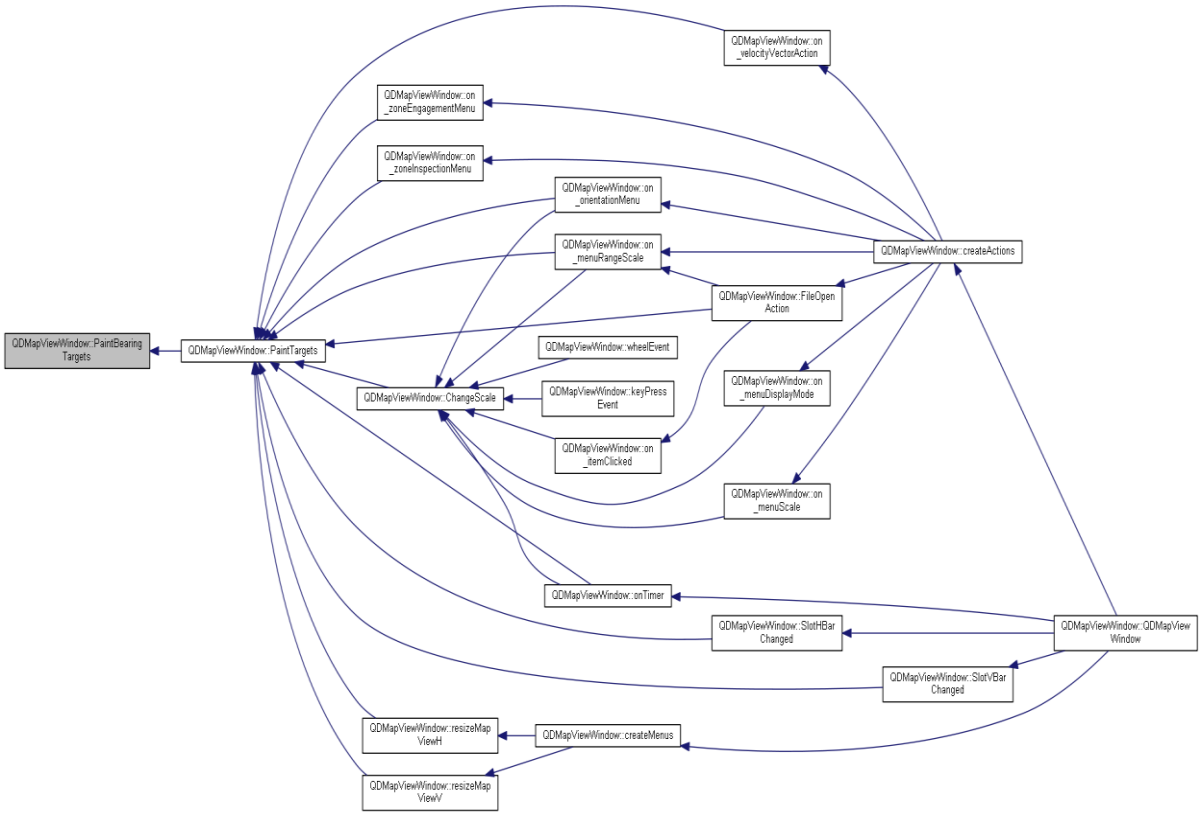

Рисунок 4.34 – Граф виклику функції PaintBearingTargets()

void PaintTargets() – функція призначена для створення графічного представлення цілей і їх промальовування на фоні карти.

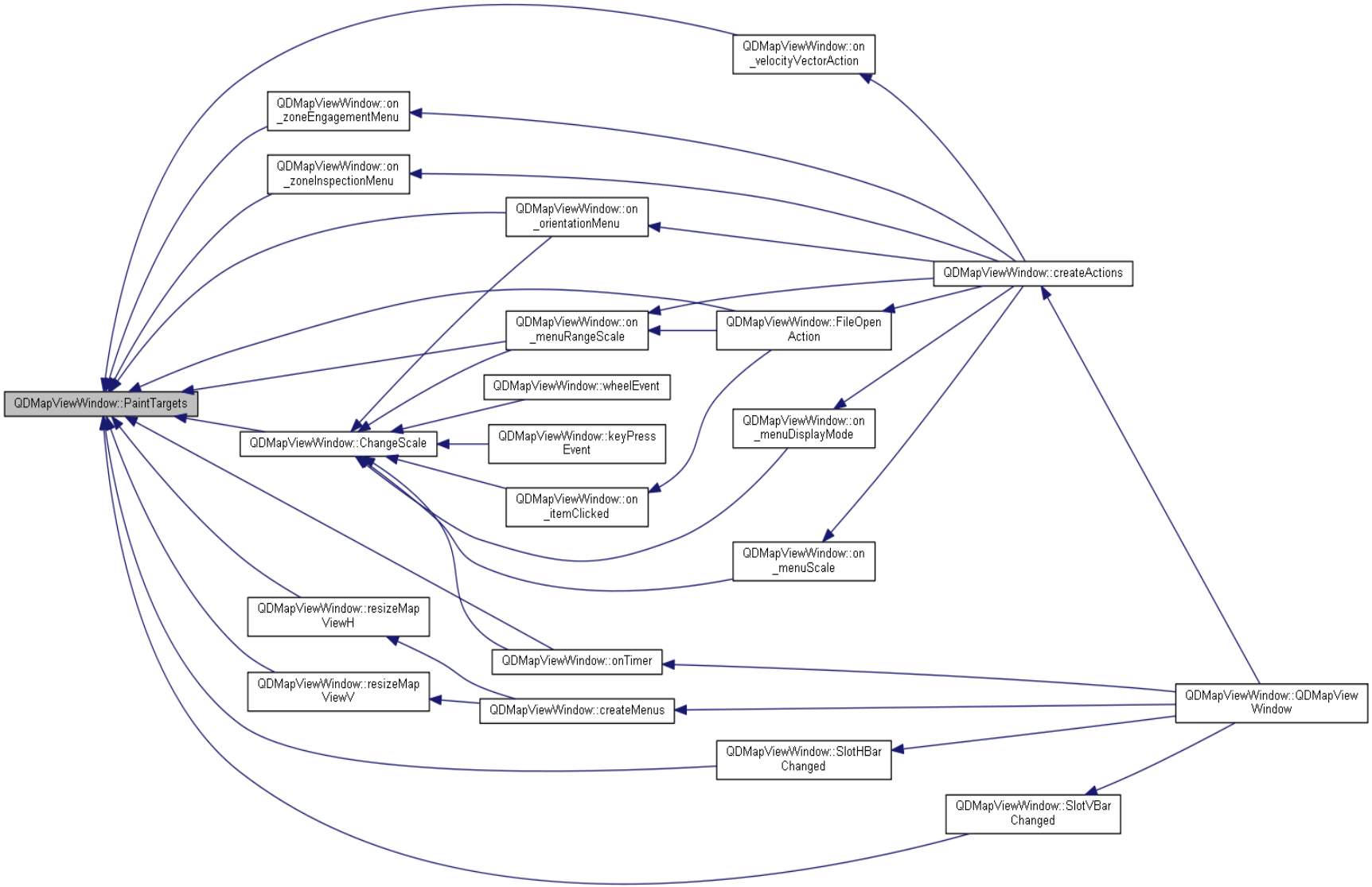

Рисунок 4.35 – Граф виклику функції PaintTargets()

void PaintZonePhoenix(), void PaintZonePozitiv(), void PaintZoneSelena(), void PaintZoneNRLS(), void PaintZonePugovitsa(),void PaintZoneMorena_A(), void PaintZoneMorena_P(), void PaintZoneProtazan(), void PaintZoneSpis(), void PaintZoneProtivodeystvie() – функції, призначені для промальовування зон освітлення сенсорів на фоні векторних карт.

void PaintZoneOtoMelara(), void PaintZoneMillennium(), void PaintZoneLafet(), void PaintZonePKRK(), void PaintZoneMhonta() – функції, призначені для промальовування зон ураження зброєю.

void readDataFusion() – функція, призначена для вичитки об'єднаних формулярів цілей зі сховища в оперативній пам'яті.

Рисунок 4.36 Граф виклику функції readDataFusion()

Void UpdateFormulars(QMap<int, QTableWidget*> *fm, QVector<FusedTargetData> *trg) – функція, призначена для оновлення положення і інформації в скороченому формулярі на фоні карти.

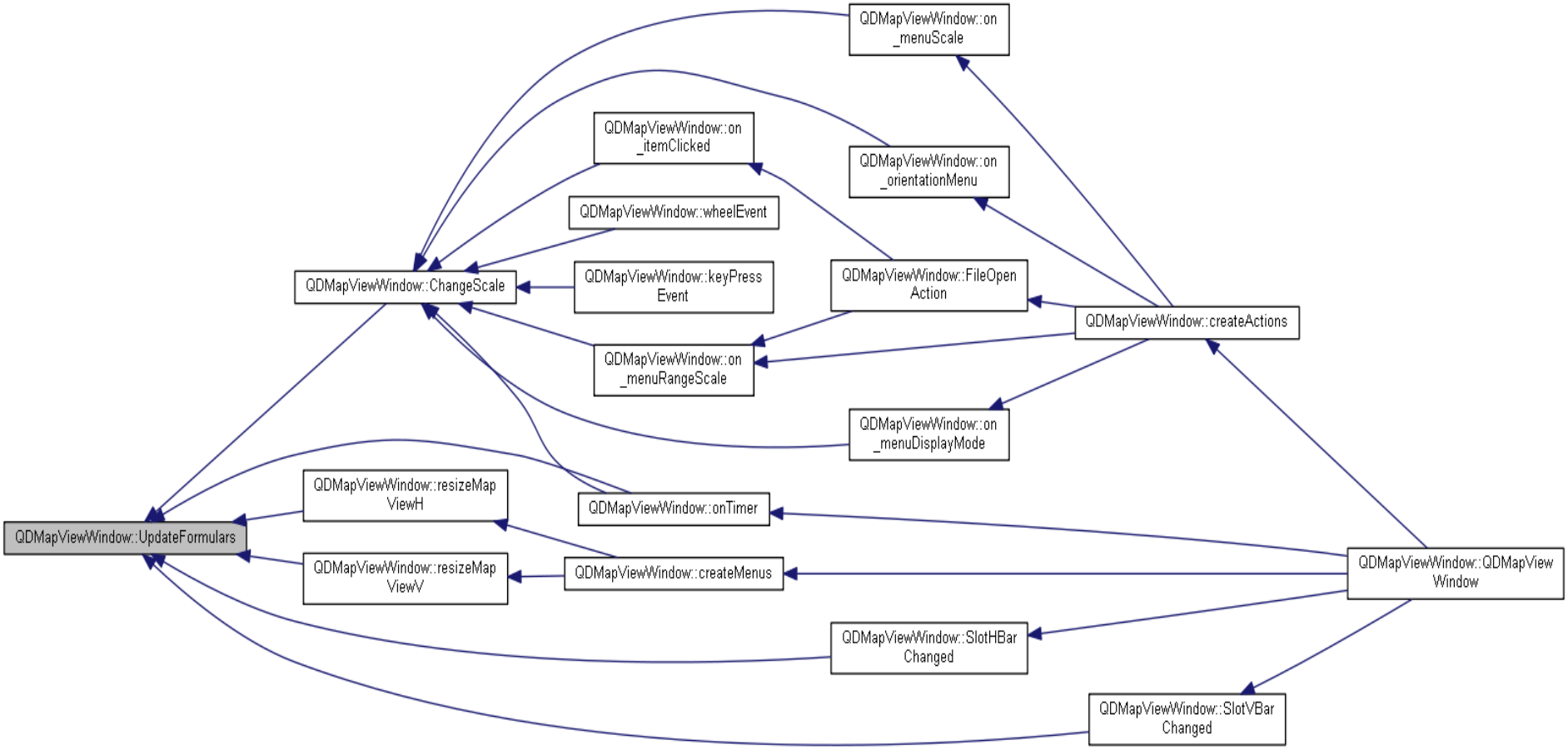

Рисунок 4.37 – Граф виклику функції UpdateFormulars(QMap<int, QTableWidget*> *fm, QVector<FusedTargetData> *trg)

4.9. Оцінка ресурсів пам’яті для програми динамічного відслідковування за надводним та повітряним просторами

– обсяг віртуальної пам'яті, займаний програмою в процесі виконання при відкритті тестової карти і відображенні тестового сценарію – приблизно 200 Мб (обсяг займаної віртуальної пам'яті може збільшитися в залежності від використовуваного функціонала, кількості відображуваних цілей);

– середній показник завантаження програмою одного логічного ядра центрального процесора при статичному відображенні статичної тестової карти і тестового сценарію – на рівні 10%. При переміщенні і масштабування карти, розширенні використовуваного функціонала навантаження даний показник може підвищуватися до 30 - 40%.

5. ОПИС ВИКОРИСТАНИХ В РОБОТІ АЛГОРИТМІВ ФУНКЦІОНУВАННЯ СИСТЕМИ

5.1. Алгоритм об’єднаної обробки треків цілей

Алгоритм об’єднаної обробки треків цілей складається з наступних частин:

– читання конфігураційного файлу;

– прийом даних про цілі з модуля прийому і декодування інформації від локаторів, встановлених на кораблі та лінії зв’язку;

– прийом даних про цілі від модулів імітаторів локаторів, встановлених на кораблі та лінії зв’язку;

– обробка призначених цілей зенітно ракетним комплексом;

– обробка цілей типу «тактична балістична ракета»;

– обробка постановника активних завад;

– ототожнення треків;

– ототожнення відміток.

Конфігураційний файл задає конфігурацію і параметри алгоритму глобальної об'єднаної обробки:

– геодезичні координати базового сенсора, щодо якого ведеться перерахунок координат (за замовчуванням – власний корабель);

– перелік сенсорів або імітаторів, інформація з яких передається в програму ототожнення;

– перелік особливо небезпечних цілей, які передаються в програми оцінки тактичної обстановки без ототожнення (балістичні, низколетючі, постановник активних завад і т.д.);

– тип алгоритму ототожнення (об'єднання треків з / без обчислення коваріаційної матриці ототожненого треку, об'єднання відміток та ін.);

– тимчасової ТВО об'єднаної обробки (фіксований, адаптивний; за замовчуванням – фіксований цикл, рівний циклу огляду 3D радіолокатора).

Алгоритм прийому даних про цілі з модуля прийому і декодування інформації від локаторів, встановлених на кораблі та лінії зв’язку формує відповідні структури даних.

Алгоритм прийому даних про цілі від модулів імітаторів локаторів, встановлених на кораблі та лінії зв’язку ініціюється в режимі «Тренаж» з паралельною обробкою реальних цілей.

Алгоритм обробки призначених цілей корабля відбирає цілі з глобальною нумерацією корабля або локальною нумерацією сенсору, за якими корабель вже веде бойові дії:

– цілі, за якими прийнято ціле вказання;

– цілі, призначені командиром корабля при автономній роботі системи.

Якщо дані отримуються через лінію зв’язку, то після селекції здійснюється перерахунок прийнятих координат від локатору, котрий передав нам дані про ціль на момент часу початку нового циклу об'єднаної обробки за формулами:

$$X_i^j = X_i^{j-1} + Vx_i^{j-1}\Delta t_i^{j-1}$$
$$Y_i^j = Y_i^{j-1} + Vy_i^{j-1}\Delta t_i^{j-1}$$
$$Z_i^j = Z_i^{j-1} + Vz_i^{j-1}\Delta t_i^{j-1}$$

(1)

де $\Delta_i^{j-1} = T^j - t_i^{j-1}$,

T^j – мітка часу початку нового циклу об’єднаної обробки,

t_i^{j-1} – мітка часу даних про i –ту ціль, отриману в j -1 циклі об’єднаної обробки.

Після часового зрівнювання виконується просторове зрівнювання – перерахунок координат з системи координат з початком в точці стояння сенсора в систему координат корабля (рисунок 3.1.1) з початком в точці стояння корабля за формулами:

$$R^m = C^{m|n} R^n + D^{m|n}, \quad (2)$$

де R^m –- вектор цілі координат в топоцентричній системі координат (ТПЦСК) корабля (координата X – схід; координата Y – північ; координата Z – вгору по нормалі до еліпсоїда WGS84), скорочено ENU (East-Nord-Up);Rn – вектор цілі координат в топоцентричній системі координат (ТПЦСК) сенсора, від якого прийшла ціль.

Елементи матриці $C^{m|n}$ обчислюються за формулами:

$$\begin{aligned}
&C^{m|n}(1,1) = \cos \Delta l^{m|n} \\
&C^{m|n}(1,2) = -\sin \phi^n \sin \Delta l^{m|n} \\
&C^{m|n}(1,3) = \cos \phi^n \sin \Delta l^{m|n} \\
&C^{m|n}(2,1) = \sin \phi^m \sin \Delta l^{m|n} \\
&C^{m|n}(2,2) = \sin \phi^m \sin \phi^n \cos \Delta l^{m|n} + \cos \phi^m \cos \phi^n \\
&C^{m|n}(2,3) = -\sin \phi^m \cos \phi^n \cos \Delta l^{m|n} + \cos \phi^m \sin \phi^n \\
&C^{m|n}(3,1) = -\cos \phi^n \sin \Delta l^{m|n} \\
&C^{m|n}(3,2) = -\cos \phi^m \sin \phi^n \cos \Delta l^{m|n} + \sin \phi^m \cos \phi^n \\
&C^{m|n}(3,3) = \cos \phi^m \cos \phi^n \cos \Delta l^{m|n} + \sin \phi^m \sin \phi^n
\end{aligned} \quad (3)$$

де l^m – довгота точки стояння корабля;

l^n – довгота точки стояння сенсора, від якого було отримано ціль;

$\Delta l^{m|n} = l^n - l^m$.

Елементи вектора $D^{m|n}$ вираховуються за формулами:

$$\begin{aligned}
&D^{m|n}(1) = (N^n + H^n) \cos \phi^n \sin \Delta l^{m|n} \\
&D^{m|n}(2) = -(N^n + H^n) \cos \phi^n \sin \phi^m \cos \Delta l^{m|n} \\
&+(N^n(1-e^2) + H^n) \cos \phi^m \sin \phi^n + N^m e^2 \cos \phi^m \sin \phi^m \\
&D^{m|n}(3) = (N^n + H^n) \cos \phi^n \cos \phi^m \cos \Delta l^{m|n} - (N^m + H^m) \\
&+(N^n(1-e^2) + H^n) \sin \phi^m \sin \phi^n + N^m e^2 (\sin \phi^m)^2
\end{aligned} \quad (4)$$

де ϕ^m – широта точки стояння корабля,

ϕ^n – широта точки стояння сенсора, від якого було отримано ціль,

$N^m = \frac{a}{\sqrt{1-e^2 \sin^2 \phi^m}}$ – радіус кривизни першого вертикалу в точці стояння корабля,

$N^n = \frac{a}{\sqrt{1-e^2 \sin^2 \phi^n}}$ – радіус кривизни першого вертикалу в точці стояння сенсора, від якого було отримано ціль,

H^m – висота точки стояння фазованого центру антени корабля,

H^n – висота точки стояння фазованого центру антени сенсора, від якого було отримано ціль,

a = 6378137 – велика напіввісь земного еліпсоїду,

e = 0,08181919.

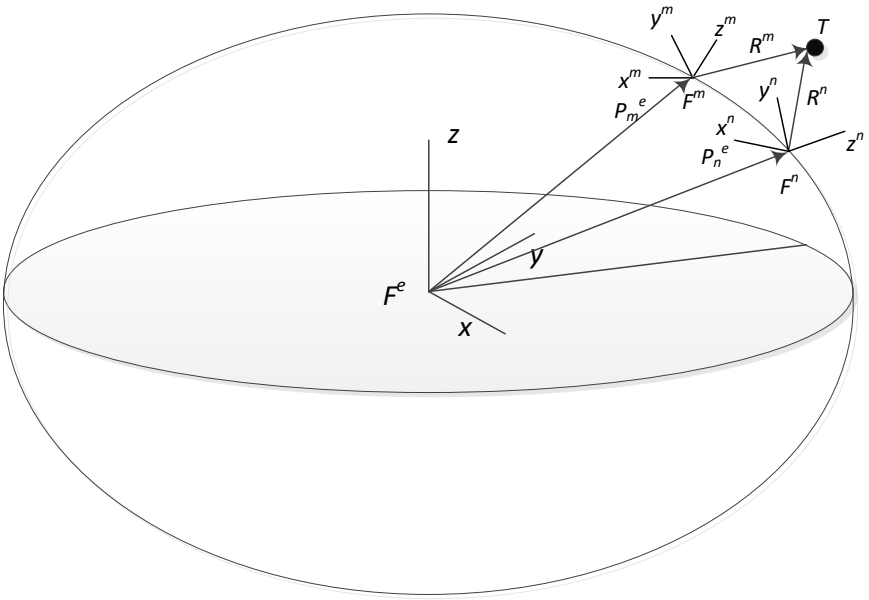

Рисунок 5.1 – Перетворення координат цілей з системи координат сенсора, від якого було отримано ціль (F^n) в систему координат корабля (F^m). F^e – геоцентрична система координат

Перетворення коваріаційної матриці P^n вектора R^n при переході до системи координат корабля виконується за формулою:

$$P^m = C^{m|n} P^n (C^{m|n})^T ,$$

(5)

а перетворення вектора швидкостей V_{R_n} вектора R^n при переході до системи координат корабля виконується за формулою:

$$V_{R_m} = C^{m|n} V_{R_n} . \quad (6)$$

5.1.1. Алгоритм обробки цілей типу оперативна тактично балістична ракета

Алгоритм обробки цілей типу оперативна тактично балістична ракета (ОТБР) також виконує тимчасове і просторове вирівнювання координат цілей за формулами (1-4).

При заданні в файлі конфігурації функції екстраполяції траєкторії польоту ОТБР з метою визначення координат, упереджених на час τ щодо поточного стану, або координат точки падіння ініціалізується алгоритм екстраполяції.

Екстраполяція проводиться в курсовій системі координат методом чисельного інтегрування системи диференціальних рівнянь плоского руху центру мас ОТБР. На рисунку 4.2 показана балістична ціль Т на ділянці низхідної інерційної траєкторії з вектором швидкості V_T і кутом тангажа γ_T. Кут тангажу може бути обчислений за формулою:

$$\gamma_T = arctg \frac{-V_H}{V_R}$$

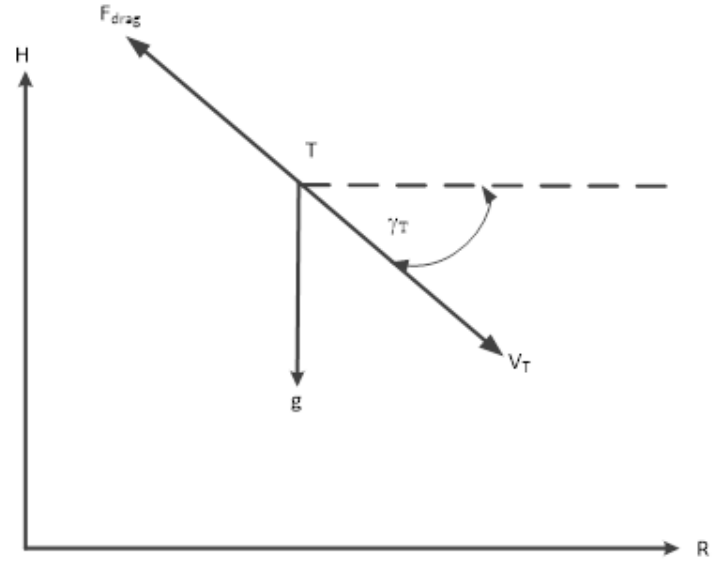

Рисунок 5.2 – Сили, котрі діють на центр мас ОТБР

Рух ОТБР може бути описаний рівняннями:

$$\frac{dV_R}{dt} = \frac{-F_{drag}}{m}\cos\gamma_T = \frac{-QS_{ref}C_{D0}g}{W}\cos\gamma_T = \frac{-Qg}{\beta}\cos\gamma_T$$
$$\frac{dV_H}{dt} = \frac{-F_{drag}}{m}\sin\gamma_T - g = \frac{QS_{ref}C_{D0}g}{W}\sin\gamma_T - g = \frac{Qg}{\beta}\sin\gamma_T - g \quad , \qquad (7)$$

де W – вага ракети,

S_{ref} – площа перетину міделю ракети,

C_{D0} – лобовий опір при нульовій підйомній силі,

β – балістичний коефіцієнт,

Q – динамічний тиск,

$\rho = \rho(H)$ – щільність повітря,

$$Q = 0.5\rho V_T^2$$

$$V_T = \sqrt{V_H^2 + V_R^2}$$

Інтегрування рівнянь (7) з початковими умовами H_0, R_0, V_0 дозволяє обчислити точку падіння ОТБР і своєчасно видати ЦВ без об'єднаної обробки. Результати розрахунку низхідної траєкторії ОТБР показані на рисунку 3.4.

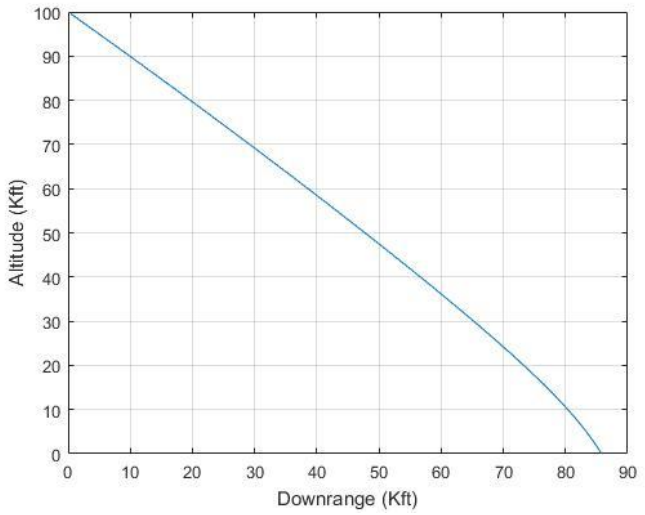

Рисунок 5.3 – Низхідна траєкторія ОТБР.

5.1.2. Алгоритм обробки для цілей типу постановник активних завад

На першому кроці здійснюється тимчасове і просторове зрівнювання кутових координат постановник активних завад (далі – ПАЗ) з умовною дальністю $D_{ПАЗ}$ = 200 км (задається в конфігураційному файлі).

На другому кроці виконується уточнення даних по ПАЗ, отриманих від кожної РЛС, що виявили ПАЗ. Для цього перебираються всі пеленги ПАЗ даної РЛС і їм присвоюються тимчасові номери. Якщо для пеленга з номером j знайдуться пеленги з номерами k_1, k_2 ... k_n, які потрапляють в інтервал (Z_{nj} - d) .. (Z_{nj} + d), то всі ці пеленги усереднюються (Z_{nj} + Z_{nk1} + ... + Z_{nkn}) / (n +1). Тут Z_{nj} – вектор пеленга відмітки з номером j, d – величина строба, в межах якого усереднюються дані, отримані від РЛС. Усереднення проводиться як по азимуту, так і по куту місця.

5.1.3. Алгоритм ототожнення треків цілей

Завдання об'єднання треків полягає в асоціації треків, що надходять від різних локальних сенсорів і перетворенні їх в глобальні системні треки за алгоритмом, наведеним на малюнках 5.4., 5.5.

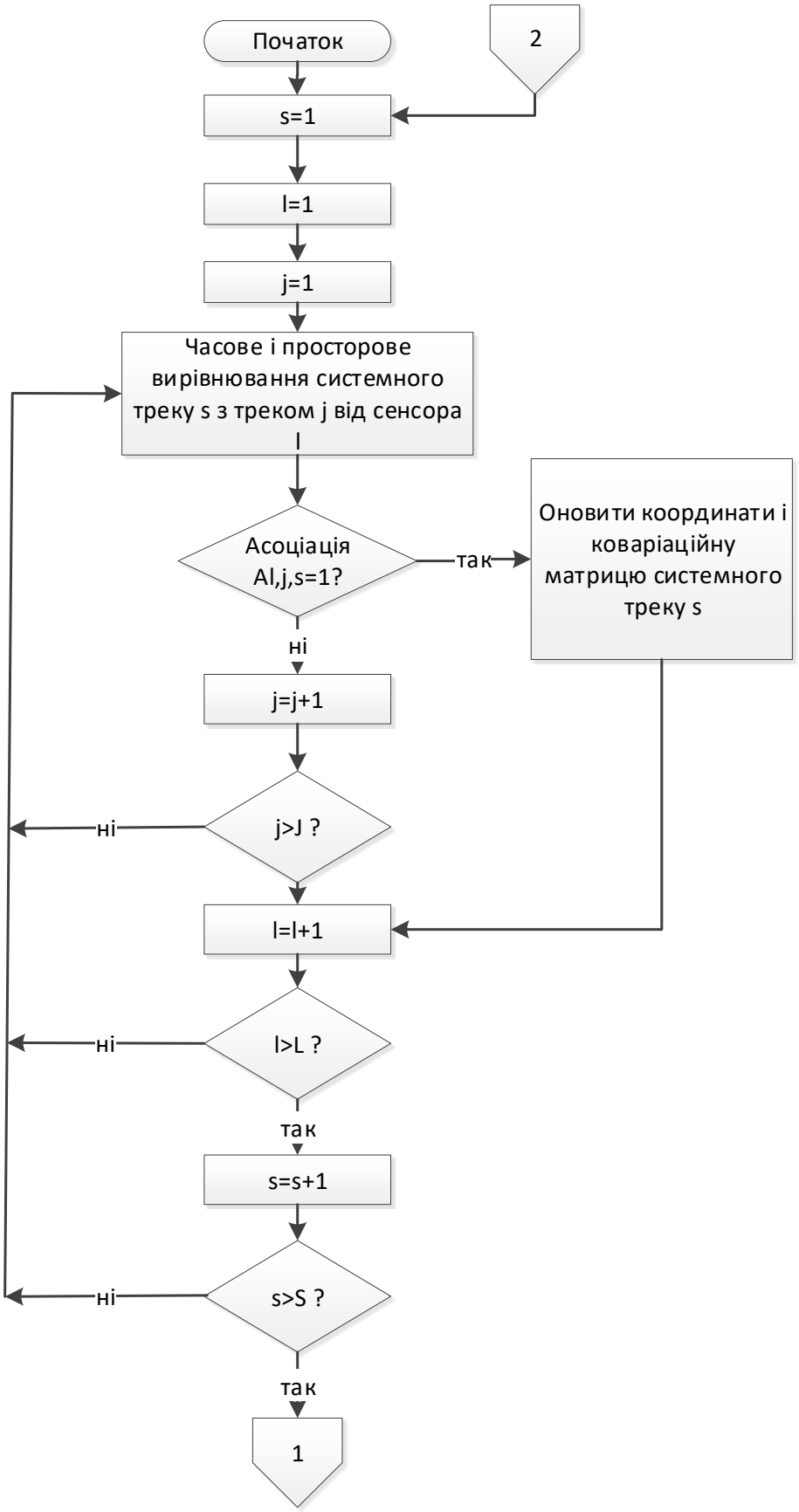

Рисунок 5.4 – Блок-схема алгоритму об'єднаної обробки треків цілей

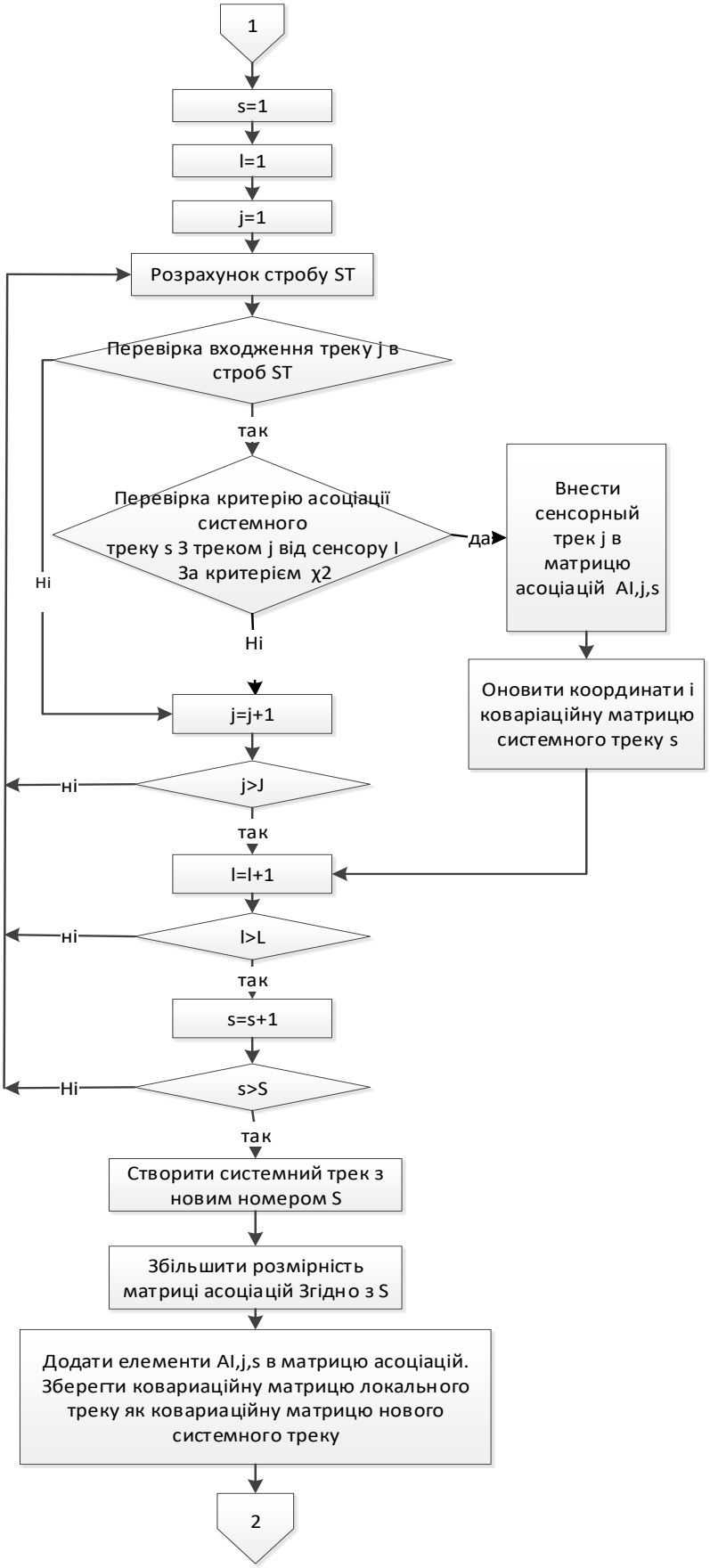

Рисунок 5.5 – Блок-схема алгоритму об'єднаної обробки треків цілей (продовження)

У процесі об'єднаної обробки виконуються наступні процедури:

1. Спочатку виконується тимчасове і просторове вирівнювання системного треку s (s = 1, ..., S) з треком j (j = 1, ..., J) від сенсора l (l = 1, ..., L) за формулами (1-6).

Також перевіряється існування асоціації треку j від сенсора l з системним треком s (перегляд таблиці асоціацій Al, j, s) за номерами системного і локального треку. Якщо буде виявлено, що немає жодного сформованого системного треку, то сенсорний трек j заноситься в таблицю асоціацій як системний з індексом s = 1, і строб ST обчислюється по середньому квадратичному відхиленню (далі – СКВ) даного системного треку.

2. Якщо вже раніше трек з локальним номером j сенсора l вже був асоційований з одним із системних треків s, то виконується оновлення координат і коваріаційної матриці системного треку s за формулами:

$$R_s^{k+1} = P_j^k (P_s^k + P_j^k)^{-1} R_S^k + P_s^k (P_s^k + P_j^k)^{-1} R_j^k \quad (8)$$

$$P_s^{k+1} = P_s^k (P_s^k + P_j^k)^{-1} P_j^k \quad (9)$$

де R_S^k – вектор стану системного треку s в момент часу k,

P_s^k – коваріаційна матриця системного треку s в момент часу k,

R_j^k – вектор стану локального треку j в момент часу k,

P_j^k – коваріаційна матриця локального треку j в момент часу k,

R_s^{k+1} – оновлений вектор стану системного трека s в момент часу k+1,

P_s^{k+1} – оновлена коваріаційна матриця системного треку s в момент часу k+1.

3. Якщо даний трек з локальним номером j раніше не був асоційований ні з одним з системних треків s, то формується строб ST розміром $3\sigma_r^k$ системного треку s і перевіряється умова:

$$\left| x_j^k - x_s^k \right| \le 3\sigma_r^k$$

$$\left|y_j^k - y_s^k\right| \le 3\sigma_r^k$$

(10)

$$\left|z_j^k - z_s^k\right| \le 3\sigma_r^k$$

де $\sigma_r^k = \sqrt{(\sigma_s^k)^2 + (\sigma_j^k)^2}$

Якщо знаходиться один або більше сенсорних треків, які відповідають умові (10), то перевіряється умова асоціації з цими сенсорними треками за критерієм χ2. Для цього обчислюється нормалізована дистанція d_{js}^2 і перевіряється умова:

$$d_{js}^2 = \frac{(x_j^k - x_s^k)^2 + (y_j^k - y_s^k)^2 + (z_j^k - z_s^k)^2}{(\sigma_r^k)^2} \le \gamma$$

(11)

Дистанція d_{js}^k є сумою квадратів M незалежних гауссових змінних з нульовими середніми і одиничними стандартними відхиленнями, тобто квадратична форма d_{js}^k має розподіл χ2, а значення γ визначається з таблиць χ2 (див. Таблицю 5.1) або по відомим апроксимаціям цієї таблиці.

Таблиця 5.1 – χ – квадрат

m/α	0,990	0,980	0,950	0,900	0,800	0,200	0,100	0,050	0,020	0,010	0,001
1	0,000	0,001	0,004	0,016	0,064	1,642	2,706	3,841	5,412	6,635	10,827
2	0,020	0,040	0,103	0,211	0,446	3,219	4,605	5,991	7,824	9,210	**13,815**
3	0,115	0,185	0,352	0,584	1,005	4,642	6,251	7,815	9,837	11,341	16,268
4	0,297	0,429	0,711	1,064	1,649	5,989	7,779	9,488	11,668	13,277	18,465
5	0,554	0,752	1,145	1,610	2,343	7,289	9,236	11,070	13,388	15,086	20,517
6	0,872	1,134	1,635	2,204	3,070	8,558	10,645	12,592	15,033	16,812	22,457
7	1,239	1,564	2,167	2,833	3,822	9,803	12,017	14,067	16,622	18,475	24,322
8	1,646	2,032	2,733	3,490	4,594	11,030	13,362	15,507	18,679	20,090	26,125
9	2,088	2,532	3,325	4,168	5,380	12,242	14,684	16,919	19,679	21,666	27,877

Приведено приклад асоціації для двомірного випадку. Нехай x_o = 100 і y_o = 200 з σ_0^2 = 16. Також x_p = 85 і y_p = 217 з σ_p^2 = 20. Тоді сумарне стандартне відхилення буде дорівнювати $\sigma_r = \sqrt{\sigma_0^2 + \sigma_p^2} = \sqrt{16+20} = 6$.

$\left|x_0 - x_p\right| = abs(100-85) = 15 = 2,5\sigma_r$ і $\left|y_0 - y_p\right| = abs(200-217) = 17 = 2,83\sigma_r$,

виходячи з цього, на компонентному рівні кожне із значень менше за $3\sigma_r$ і умова асоціації задовольняється.

Після цього вирахуємо нормалізовану дистанцію $d^2 = \frac{(2,5\sigma_r)^2 + (2,83\sigma_r)^2}{\sigma_r^2} = 14,26$.

З таблиць можна побачити, що розраховане значення дистанції 14,26 при ступені свободи 2 перевищує теоретичну дистанцію 13,815 відповідну ймовірності 0.001 (0.1%), тобто умову точного стробування не задоволено і асоціація не може бути здійснена.

Очевидно, що груба перевірка на компонентному рівні не є достатньою умовою.

4. Якщо критерій виконується, то трек j сенсора l включається в матрицю асоціацій і оновлюються координати і коваріаційна матриця асоційованого системного треку s за формулами (8-9).

5. Якщо ж список кандидатів виявиться порожнім, то створюється новий системний трек s. Тоді необхідно збільшити розмірність матриці асоціацій до S + 1 і додати елемент AI, j, s у матрицю асоціацій. При цьому коваріаційну матрицю локального треку необхідно зберегти як коваріаційну матрицю нового системного треку.

Варіант ототожнення треків може бути реалізований і в спрощеному вигляді без обчислення коваріаційних матриць і поновлення системних ототожнених треків шляхом «відкидання» менш точних треків.

5.2. Алгоритми маневрування

Для поліпшення якості дій капітана корабля та виключення можливостей людської помилки в системі розроблені такі алгоритми маневрування :

- утримання заданої позиції;
- зміна дистанції до цілі в найкоротші строки;
- зближення впритул;
- зміна дистанції без зміни пеленга;
- зміна пеленга без зміни відстані;
- вихід на мінімальну відстань по носу чи на максимальну відстань за кормою.

5.2.1. Алгоритм утримання заданої позиції

Одним з найважливіших та найрозповсюдженіших алгоритмів маневрування, є утримання заданої позиції. На відміну від інших алгоритмів, він використовуються при русі корабля в ордері чи тактичній групі [9]. Для виконання маневру потрібно змінити свій курс на новий курс, який задається зрівнювачем. В точці закінчення маневру потрібно зберегти відстань і пеленг на зрівнювач таким, як і до зміни курсу.

Вхідні дані:

- відстань до цілі (target->Distance, D);
- пеленг на ціль (target->bearingAngle, $П_М$);
- курс цілі (target->course, $К_К$);
- швидкість цілі (target->v, $V_К$);
- швидкість власного корабля (MainShip_V, $V_М$);
- новий курс зрівнювача (newCourse, $К_К$).

Шукані дані:

– кут повороту зрівнювача (rotationAngle);

– кут зсуву (movingAngle);

– курс маневру (maneuveringCourse);

– відстань маневру (maneuveringDistance);

– час виконання маневру (maneuveringTime).

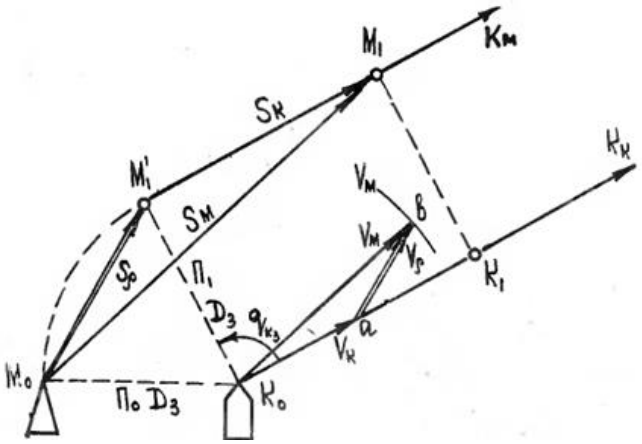

Рисунок 5.6 – Укрупнена блок-схема до алгоритму утримання заданої позиції

Розв’язок задачі:

1. Розрахунок кута повороту зрівнювача:
(rotationAngle): $|К_П - К_К|$.
2. Розрахунок кута зсуву:
movingAngle = (180 – rotationAngle) / 2.
3. Розрахунок курсу маневру, якщо курс зрівнювача менше нового курсу: maneuveringCourse = $К_К$ – movingAngle, в іншому випадку:
maneuveringCourse = $К_К$ + movingAngle.
4. Розрахунок відстані маневру:
maneuveringDistance= 2 * D * cos(movingAngle).
5. Розрахунок часу виконання маневру:
maneuveringTime = maneuveringDistance/$V_М$.

5.2.2. Алгоритм зміни дистанції до цілі в найкоротші строки

Вхідні дані:

– відстань до цілі (target->Distance, D);

– пеленг на ціль (target->bearingAngle, $П_М$);

– курс цілі (target->course, $К_К$);

– швидкість цілі (target->v, $V_К$);

– швидкість власного корабля (MainShip_V, $V_М$);

– задана дистанція (setDistance, D_1).

Шукані дані:

– курс власного корабля на точку перетину шляху з ціллю (maneuveringCourse, $К_М$);

– час до зближення на задану відстань до цілі (maneuveringTime);

– дистанція до точки перетину шляху з ціллю (maneuveringDistance, $S_М$).

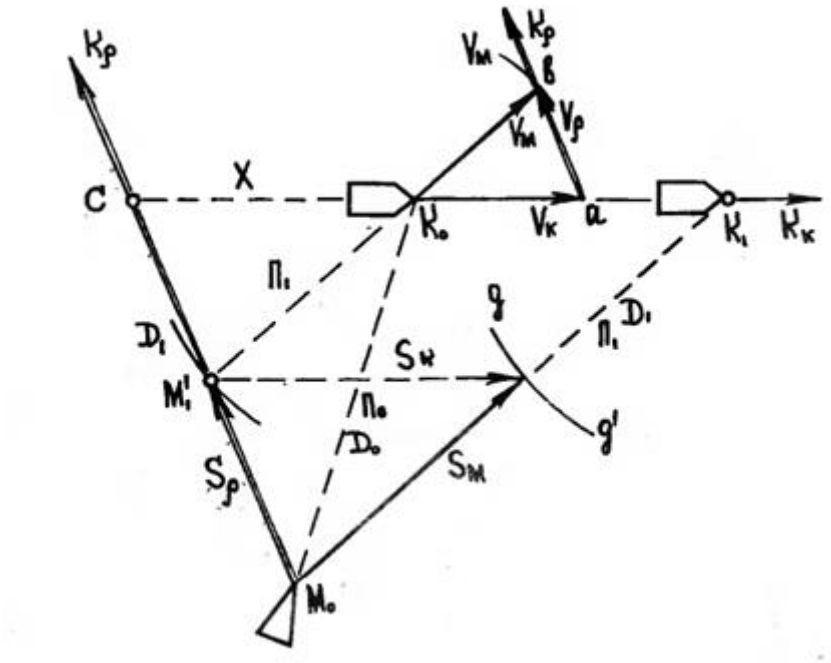

Рисунок 5.7 – Укрупнена блок-схема до алгоритму зміни дистанції до цілі в найкоротші строки

Ціль маневру – К, в даний момент часу знаходитися в початковій точці $К_0$, курс цілі – $К_К$, вектор швидкості – $V_К$. Головний корабель в даний момент часу знаходиться в початковій точці $М_0$, відстань до цілі – пряма $К_0М_0$, довжиною D_0, пеленгом $П_М$.

Для виконання даного маневру потрібно вийти на задану відстань (D_1) з курсовим кутом по закінченню рівним нулю.

Рішення даного алгоритму здійснюється за допомогою подібних трикутників: $М_0К_1С$, $М_1`К_0С$ і $М_0М_1М_1`$.

Розв’язок задачі:

1. Розрахунок Х як відношення швидкостей, помножене на задану відстань:

   $X = (V_К/V_М)*D_1$;

2. Знаходимо пеленг противника $П_К$ (targetBearingAngle). Якщо пеленг нашого корабля більше за 180 градусів:

   $П_К = П_М – 180$, в іншому випадку: $П_К = П_М + 180$;

3. Розрахунок кута $СК_0М_0$:

   reverseAngle = $||П_К$ (targetBearingAngle) – Кк| – 180|;

4. Розрахунок відстані $СМ_0$ згідно з теоремою косинусів:

   relativeDistance = (reverseDistance^2 + D_0^2 – (2 * reverseDistance * D_0 * cos(reverseAngle)))^1/2;

5. Розрахунок кута $М_0СК_1$. Якщо $СК_0М_0$(reverseAngle) < 90:

   relativeAngle = |180-|arcsin(D_0 * sin(reverseAngle))||, в іншому випадку: relativeAngle = |arcsin(D_0 * sin(reverseAngle))|.

6. Розрахунок кута $СМ_0К_1$:

   relativeToManeuveringAngle = |arcsin(D_0 * sin(reverseAngle))|.

7. Розрахунок кута $СК_1М_0$:

   enemyToManeuveringAngle = |180 – relativeToManeuveringAngle($СМ_0К_1$) – relativeAngle($М_0СК_1$)|;

8. Розрахунок відстані $М_0К_1$:

fullDistance = relativeDistance(CM_0) * sin(relativeAngle(M_0CK_1)) / sin(enemyToManeuveringAngle(CK_1M_0));

9. Розрахунок кута $K_0M_0K_1$:

bearingToManeuveringAngle = |180 - enemyToManeuveringAngle(CK_1M_0) - (180 - reverseAngle(CK_0M_0))|;

10. Обчислення курсу власного корабля змінюється в залежності від розташування цілі та її курсу, всього можливо 24 варіанта, які зводяться до 4 ситуацій в залежності від того, в якій чверті знаходиться ціль.

1. Якщо $0 <= П_М < 90$ (перша чверть):
   - якщо $|К_К - П_М| < 90$ & $К_К < П_К$ & $0 <= К_К < 90$;
   - якщо $270 <= К_К < 360$ & $180 <= П_К < 270$;
   - якщо $|К_К - П_М| < 90$ & $К_К > П_К$ & $180 <= К_К < 270$.

Якщо хоча б одна з перевірок істинна, то:

$К_М = П_К -$ bearingToManeuveringAngle($K_0M_0K_1$), в іншому випадку: $К_М = П_К +$ bearingToManeuveringAngle($K_0M_0K_1$).

2. Якщо $90 <= П_М < 180$ (друга чверть):
   - якщо $|К_К - П_М| < 90$ і $К_К < П_К$ і $90 <= К_К < 180$;
   - якщо $0 <= К_К < 90$ і $270 <= П_К < 360$;
   - якщо $|К_К - П_М| < 90$ і $К_К > П_К$ иі$270 <= К_К < 360$.

Якщо хоча б одна з перевірок істинна, то:

$К_М = П_К -$ bearingToManeuveringAngle($K_0M_0K_1$), в іншому випадку: $К_М = П_К +$ bearingToManeuveringAngle($K_0M_0K_1$).

3. Якщо $180 <= П_М < 270$ (третя чверть):
   - якщо $|К_К - П_М| < 90$ & $К_К > П_К$ & $180 <= К_К < 270$;
   - якщо $270 <= К_К < 360$ & $0 <= П_К < 90$;
   - якщо $|К_К - П_М| < 90$ & $К_К < П_К$ & $0 <= К_К < 90$.

Якщо хоча б одна з перевірок істинна, то:

$К_М = П_К$ + bearingToManeuveringAngle($К_0М_0К_1$), в іншому випадку: $К_М = П_К$ – bearingToManeuveringAngle($К_0М_0К_1$).

4. Якщо 270 <= $П_М$ < 360 (четверта чверть):

   – якщо $|К_К – П_М| < 90$ & $К_К > П_К$ & 270 <= $К_К$ < 360;

   – якщо 0 <= $К_К$ < 90 & 90 <= $П_К$ < 180;

   – якщо $|К_К – П_М| < 90$ & $К_К < П_К$ & 90 <= $К_К$ < 180.

Якщо хоча б одна з перевірок істинна, то:

$К_М = П_К$ + bearingToManeuveringAngle($К_0М_0К_1$), в іншому випадку: $К_М = П_К$ – bearingToManeuveringAngle($К_0М_0К_1$).

11. Розрахунок відстані($S_М$) до точки($М_1$) закінчення маневру: maneuveringDistance = fullDistance($М_0К_1$) – D_0(SetDistance);

12. Розрахунок часу, який потрібен для здійснення маневру: maneuveringTime = maneuveringDistance($S_М$)/MainShip_V($V_М$).

5.2.3. Алгоритм зближення впритул

Вхідні дані:

– дистанція до цілі (target->Distance, D);

– пеленг на ціль (target->bearingAngle, $П_М$);

– курс цілі (target->course, $К_К$);

– швидкість цілі (target->v, $V_К$);

– швидкість власного корабля (MainShip_V, $V_М$).

Шукані дані:

– курс власного корабля на точку перетину шляху з ціллю (maneuveringCourse, $К_М$);

– час до зближення впритул з ціллю (maneuveringTime);

– дистанцію до точки перетину шляху з ціллю (maneuveringDistance, $S_М$).

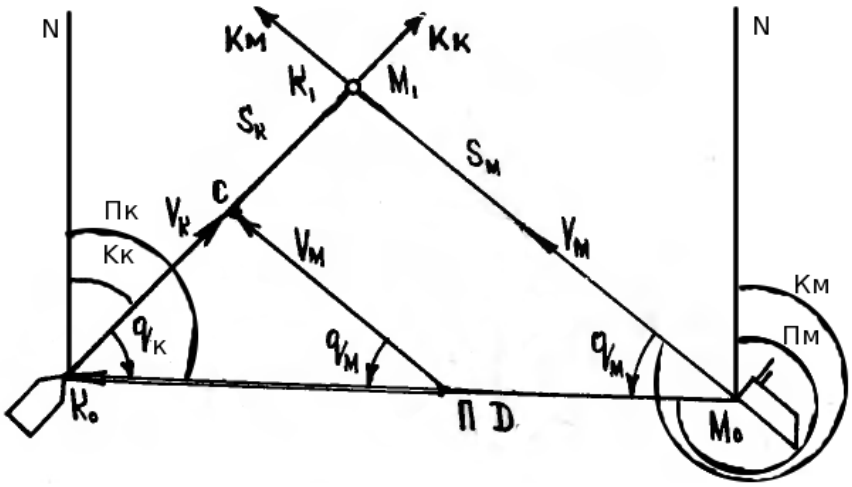

Рисунок 5.8 – Укрупнена блок-схема до алгоритму зближення впритул

Аналітичне рішення задачі засноване на побудові трикутників швидкостей (див. Рис. 5.8).

Ціль маневру – К, в даний момент часу знаходитися в початковій точці $К_0$, курс цілі– $К_К$, вектор швидкості – $V_К$. Власний корабель в даний момент часу знаходиться в початковій точці $М_0$, відстань до цілі– пряма $К_0М_0$, відстанню D, пеленгом $П_М$, і вектором швидкості $V_М$.

При постійній швидкості власного корабля і об'єкта, а також при незмінному курсі об'єкта, кораблі зустрінуться в кінцевій точці $К_1$ ($М_1$).

Розв’язок задачі:

Для знаходження курсу зближення потрібно дізнатися курсовий кут власного корабля (qм), для цього потрібно побудувати трикутник одиничних векторів швидкостей на сторонах курсу об'єкта і відстані до об'єкта від власного корабля. Так як трикутники $К_0СП$ і $К_0К_1М_0$ подібні (за двома сторонами і кутом між ними), то кути трикутників рівні.

1. Знаходимо пеленг супротивника на власний корабель $П_К$ (targetBearingAngle). Якщо пеленг власного корабля більший за 180 градусів: $П_К = П_М - 180$, в іншому випадку: $П_К = П_М + 180$.

2. Знаходимо курсовий кут супротивника $q_К$ (targetCourseAngle).

Для обчислення даного кута потрібно спочатку визначити в якій чверті щодо власного корабля перебуває ціль маневру. Якщо ціль знаходиться в першій або другій чвертях, то необхідно здійснити наступні перевірки:

– якщо $|К_К - П_М| < 90$ & $К_К < П_К$;

– якщо $0 <= К_К < 90$ & $270 <= П_К < 360$.

Якщо хоча б одна з перевірок істинна, то:

$q_К = 360 - |П_К - К_К|$, в іншому випадку: $q_К = |П_К - К_К|$.

Якщо ціль знаходиться в третій або четвертій чвертях, то необхідно здійснити наступні перевірки:

– якщо $|К_К - П_М| < 90$ & $К_К > П_К$;

– якщо $270 <= К_К < 360$ & $0 <= П_К < 90$.

Якщо хоча б одна з перевірок істинна, то:

$q_К = 360 - |П_К - К_К|$, в іншому випадку: $q_К = |П_К - К_К|$.

3. Знаходимо курсовий кут власного корабля $q_М$(mainCourseAngle) згідно з теоремою синусів:

$q_М = \arcsin((V_К * \sin(q_К) / V_М))$.

4. Обчислення курсу власного корабля змінюється в залежності від розташування цілі та її курсу, всього можливо 24 варіанта, які зводяться до 4 ситуацій в залежності від того, в якій чверті знаходиться ціль:

1. Якщо $0 <= П_М < 90$ (перша чверть):

– якщо $|К_К - П_М| < 90$ & $К_К < П_К$ & $0 <= К_К < 90$;

– якщо $270 <= К_К < 360$ & $180 <= П_К < 270$;

– якщо $|К_К - П_М| < 90$ & $К_К > П_К$ & $180 <= К_К < 270$.

Якщо хоча б одна з перевірок істинна, то:

$К_М = П_К - q_М$, в іншому випадку: $К_М = П_К + q_М$.

2. Якщо $90 <= П_М < 180$ (друга чверть):

– якщо $|К_К - П_М| < 90$ і $К_К < П_К$ і $90 <= К_К < 180$;

– якщо 0 <= $К_К$ < 90 і 270 <= $П_К$ < 360;

– якщо |$К_К$ – $П_М$| < 90 і $К_К$ > $П_К$ і 270 <= $К_К$ < 360.

Якщо хоча б одна з перевірок істинна, то:

$К_М = П_К – q_М$, в іншому випадку: $К_М = П_К + q_М$.

3. Якщо 180 <= $П_М$ < 270 (третя чверть):

– якщо |$К_К$ – $П_М$| < 90 & $К_К$ > $П_К$ & 180 <= $К_К$ < 270;

– якщо 270 <= $К_К$ < 360 & 0 <= $П_К$ < 90;

– якщо |$К_К$ – $П_М$| < 90 & $К_К$ < $П_К$ & 0 <= $К_К$ < 90.

Якщо хоча б одна з перевірок істинна, то:

$К_М = П_К + q_М$, в іншому випадку: $К_М = П_К – q_М$.

4. Якщо 270 <= $П_М$ < 360 (четверта чверть):

– якщо |$К_К$ - $П_М$| < 90 & $К_К$ > $П_К$ & 270 <= $К_К$ < 360;

– якщо 0 <= $К_К$ < 90 & 90 <= $П_К$ < 180;

– якщо |$К_К$ - $П_М$| < 90 & $К_К$ < $П_К$ & 90 <= $К_К$ < 180.

Якщо хоча б одна з перевірок істинна, то:

$К_М = П_К + q_М$, в іншому випадку: $К_М = П_К – q_М$.

5. Розрахомуємо кут зближення впритул $q_{К1}$:

(convergenceAngle) = 180 – $q_К$ – $q_М$.

6. Розрахомуємо відстань до точки зближення впритул $S_М$ (maneuveringDistance) за теоремою синусів:

$S_М = D * \sin(q_К) / \sin(q_{К1})$.

7. Розрахомуємо час, який потрібен для виконання маневру зближення впритул: (maneuveringTime) = $S_М / V_М$.

5.2.4. Зміна дистанції без зміни пеленгу

Вхідні дані:

– дистанція до цілі (target->Distance, D);

– пеленг на ціль (target->bearingAngle, $П_М$);

– курс цілі (target->course, $К_К$);

– швидкість цілі (target->v, $V_К$);

– швидкість власного корабля (MainShip_V, $V_М$);

– задана дистанція (setDistance, D_1).

Шукані дані:

– курс власного корабля на точку перетину шляху з ціллю (maneuveringCourse, $К_М$);

– час до зближення впритул з метою (maneuveringTime);

– дистанцію до точки перетину з метою (maneuveringDistance, $S_М$).

Для виконання даного маневру потрібно вийти на задану відстань (SetDistance), а також по закінченню маневру зберегти поточний пеленг.

Розв’язок задачі:

1. Знаходимо пеленг супротивника на власний корабель $П_К$ (targetBearingAngle). Якщо пеленг власного корабля більше 180 градусів:

$П_К$ (targetBearingAngle) = $П_М$ – 180, в іншому випадку: $П_К$ ((targetBearingAngle)) = $П_М$ +180;

2. Знаходимо курсовий кут цілі $Q_К$ (targetCourseAngle).

Для обчислення даного кута потрібно спочатку визначити, в якій чверті, відносно власного корабля перебуває ціль маневру. Якщо ціль знаходиться в першій або другій чвертях, то необхідно здійснити наступні перевірки:

– якщо $|К_К - П_М| < 90$ & $К_К < П_К$;

– якщо $0 <= К_К < 90$ & $270 <= П_К < 360$.

Якщо хоча б одна з перевірок істинна, то:

$q_К = 360 - |П_К - К_К|$, в іншому випадку: $q_К = |П_К - К_К|$.

Якщо ціль знаходиться в третій або четвертій чвертях, то необхідно здійснити наступні перевірки:

– якщо $|К_К - П_М| < 90$ & $К_К > П_К$;

– якщо $270 <= К_К < 360$ & $0 <= П_К < 90$.

Якщо хоча б одна з перевірок істинна, то:

$q_К = 360 – |П_К – К_К|$, в іншому випадку: $q_К = |П_К – К_К|$;

3. Розраховуємо курсовий кут власного корабля $q_К$:

mainShipCourseAngle = arcsin($V_К$ * sin(targetCourseAngle));

4. Розраховуємо курсовий кут цілі:

targetToMainCourseAngle = 180 – targetCourseAngle – mainShipCourseAngle;5. Розраховуємо relativeDistance = |D - SetDistance|;

5. Розраховуємо відстань до точки закінчення маневру:

maneuveringDistance = (relativeDistance * sin(targetCourseAngle) / sin(targetToMainCourseAngle));

6. Розраховуємо час виконання маневру:

maneuveringTime = maneuveringDistance / $V_М$;

7. Обчислення курсу власного корабля змінюється в залежності від розташування цілі і її курсу, всього можливо 24 варіанта, які зводяться до 4 ситуацій в залежності від того, в якій чверті знаходиться ціль:

1. Якщо $0 <= П_М < 90$ (перша чверть):

– якщо $|К_К – П_М| < 90$ & $К_К < П_К$ & $0 <= К_К < 90$;

– якщо $270 <= К_К < 360$ & $180 <= П_К < 270$;

– якщо $|К_К – П_М| < 90$ & $К_К > П_К$ & $180 <= К_К < 270$.

Якщо хоча б одна з перевірок істинна, то:

$К_М = П_К –$ mainShipCourseAngle, в іншому випадку: $К_М = П_К +$ mainShipCourseAngle.

2. Якщо $90 <= П_М < 180$ (друга чверть):

– якщо $|К_К – П_М| < 90$ & $К_К < П_К$ & $90 <= К_К < 180$;

– якщо $0 <= К_К < 90$ & $270 <= П_К < 360$;

– якщо $|К_К – П_М| < 90$ & $К_К > П_К$ & $270 <= К_К < 360$.

Якщо хоча б одна з перевірок істинна, то:

$К_М = П_К$ – mainShipCourseAngle, в іншому випадку: $К_М = П_К$ + mainShipCourseAngle.

3. Якщо $180 <= П_М < 270$ (третя чверть):
   - якщо $|К_К – П_М| < 90$ & $К_К > П_К$ & $180 <= К_К < 270$;
   - якщо $270 <= К_К < 360$ & $0 <= П_К < 90$;
   - якщо $|К_К – П_М| < 90$ & $К_К < П_К$ & $0 <= К_К < 90$.

Якщо хоча б одна з перевірок істинна, то:

$К_М = П_К$ + mainShipCourseAngle, в іншому випадку: $К_М = П_К$ - mainShipCourseAngle.

4. Якщо $270 <= П_М < 360$ (четверта чверть):
   - якщо $|К_К – П_М| < 90$ & $К_К > П_К$ & $270 <= К_К < 360$;
   - якщо $0 <= К_К < 90$ & $90 <= П_К < 180$;
   - якщо $|К_К – П_М| < 90$ & $К_К < П_К$ & $90 <= К_К < 180$.

Якщо хоча б одна з перевірок істинна, то:

$К_М = П_К$ + mainShipCourseAngle, в іншому випадку: $К_М = П_К$ – mainShipCourseAngle.

5.2.5. Алгоритм зміни пеленгу без зміни дистанції

Вхідні дані:

- дистанція до цілі (target->Distance, D);
- пеленг на ціль (target->bearingAngle, $П_М$);
- курс цілі (target->course, $К_К$);
- швидкість цілі (target->v, $V_К$);
- швидкість власного корабля (MainShip_V, $V_М$);
- заданий пеленг (newBearingAngle).

Шукані дані:

- курс власного корабля на точку закінчення маневру

(newBearingAngle);

– час до закінчення маневру (maneuveringTime);

– дистанція до закінчення маневру (maneuveringDistance).

Для вирішення даного алгоритму потрібно змінити поточний пеленг на новий не змінюючи відстані до цілі (D).

Розв’язок задачі:

1. Розраховуємо кут повороту:

rotationAngle = | $П_К$ – newBearingAngle |;

2. Розраховуємо кут руху:

movingAngle = (180 – rotationAngle) / 2;

3. Розраховуємо курс маневру, якщо поточний пеленг ($П_К$) менше заданого пеленга (newBearingAngle):

maneuveringAngle = $П_К$ – movingAngle, в іншому випадку: maneuveringAngle = $П_К$ + movingAngle;

4. Обчислюємо відстань маневру:

maneuveringDistance = 2 * D * cos (movingAngle);

5. Обчислюємо час маневрування:

maneuveringTime = maneuveringDistance / $V_М$;

5.2.6. Алгоритм виходу на мінімальну відстань по носу чи на максимальну відстань за кормою

Вхідні дані:

– дистанція до цілі (target->Distance, D);

– пеленг на ціль (target->bearingAngle, $П_М$);

– курс цілі (target->course, $К_К$);

– швидкість цілі (target->v, $V_К$);

– швидкість власного корабля (MainShip_V, $V_М$).

Шукані дані:

– курс власного корабля на точку закінчення маневру (newBearingAngle);

– час до закінчення маневру (maneuveringTime);

– дистанція до закінчення маневру (maneuveringDistance).

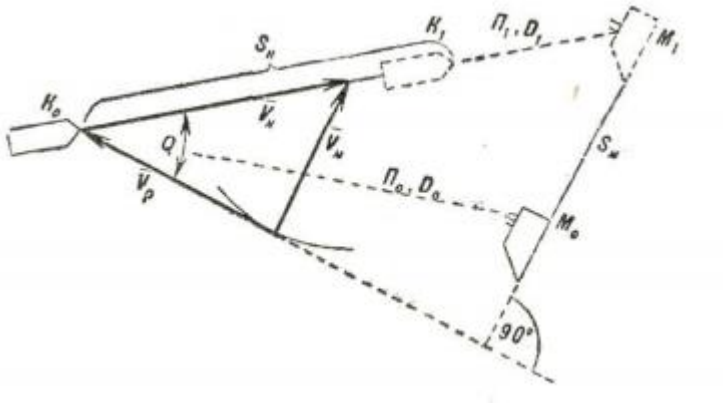

Рисунок 4.9 – Укрупнена блок-схема до алгоритму виходу на мінімальну відстань по носу

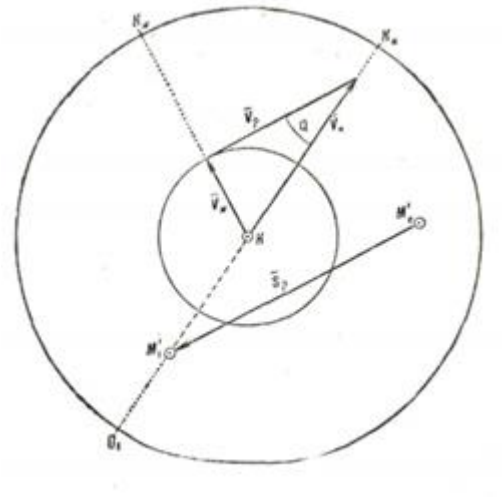

Рисунок 4.10 – Укрупнена блок-схема до алгоритму виходу на максимальну відстань за кормою.

Для виконання цього маневру ми повинні вийти на нову позицію так, щоб курсовий кут цілі дорівнював 0 або 180 градусам.

Розв’язок задачі:

1. Знаходимо пеленг цілі на власний корабель $П_К$ (targetBearingAngle). Якщо пеленг власного корабля більше 180 градусів:

$П_К$ (targetBearingAngle) = $П_М$ – 180, в іншому випадку: $П_К$

(targetBearingAngle) = $П_М$ +180;

2. Знаходимо критичний курсовий кут Q:

criticalCourseAngle = arcsin ($V_М$ / $V_К$);

3. Обчислення критичного кута і кута руху залежить від того, в якій чверті знаходиться ціль:

Якщо ціль знаходиться в першій або другій чвертях, то необхідно здійснити наступні перевірки:

- різниця пеленгу та курсу цілі < 90 & курс цілі < пеленга;
- 0 < курс цілі < 90 & 270 < пеленг < 360, то

courseAngle = 360 –| targetBearingAngle – $К_К$|;

movingAngle = 180 – criticalCourseAngle – courseAngle,

в іншому випадку:

courseAngle = |$К_К$– targetBearingAngle|; movingAngle = 180 – criticalCourseAngle – (180 – courseAngle).

Якщо ціль знаходиться в третій або четвертій чвертях, то необхідно здійснити наступні перевірки:

- різниця пеленгу та курсу цілі < 90 & курс цілі > пеленга;
- 270 < курс цілі < 360 & 0 < пеленг < 90,

то:

courseAngle = 360 –| targetBearingAngle – $К_К$|

movingAngle = 180 – criticalCourseAngle – courseAngle,

в іншому випадку:

courseAngle = |$К_К$– targetBearingAngle|;

movingAngle = 180 – criticalCourseAngle – (180 – courseAngle).

4. Розрахунок відстані, часу і курсу маневру залежить від порівняння критичного курсового кута і курсового кута:

1. критичний курсовий кут> курсовий кут,

maneuveringDistance = (D * sin (courseAngle)) /

sin (90-criticalCourseAngle);

maneuveringTime = maneuveringDistance / V_M;

2. критичний курсовий кут <курсовий кут,

maneuveringDistance = (D * sin (courseAngle)) /
sin (criticalCourseAngle);

maneuveringTime = maneuveringDistance / V_M,

3. якщо ціль знаходиться по правому борту:
maneuveringCourse = | $К_К$ – movingAngle |,

4. якщо по лівому:
maneuveringCourse = $К_К$ + movingAngle,

5. якщо ціль знаходиться по правому борту:
maneuveringCourse = $К_К$ + criticalCourseAngle – 90,

6. якщо по лівому:
maneuveringCourse = $К_К$ – criticalCourseAngle + 90.

6. ГРАФІЧНІ ІНТЕРФЕЙСИ СИСТЕМИ

Загальний вигляд графічного інтерфейсу оператора надводної та повітряної обстановок представлений на рисунку 6.1.

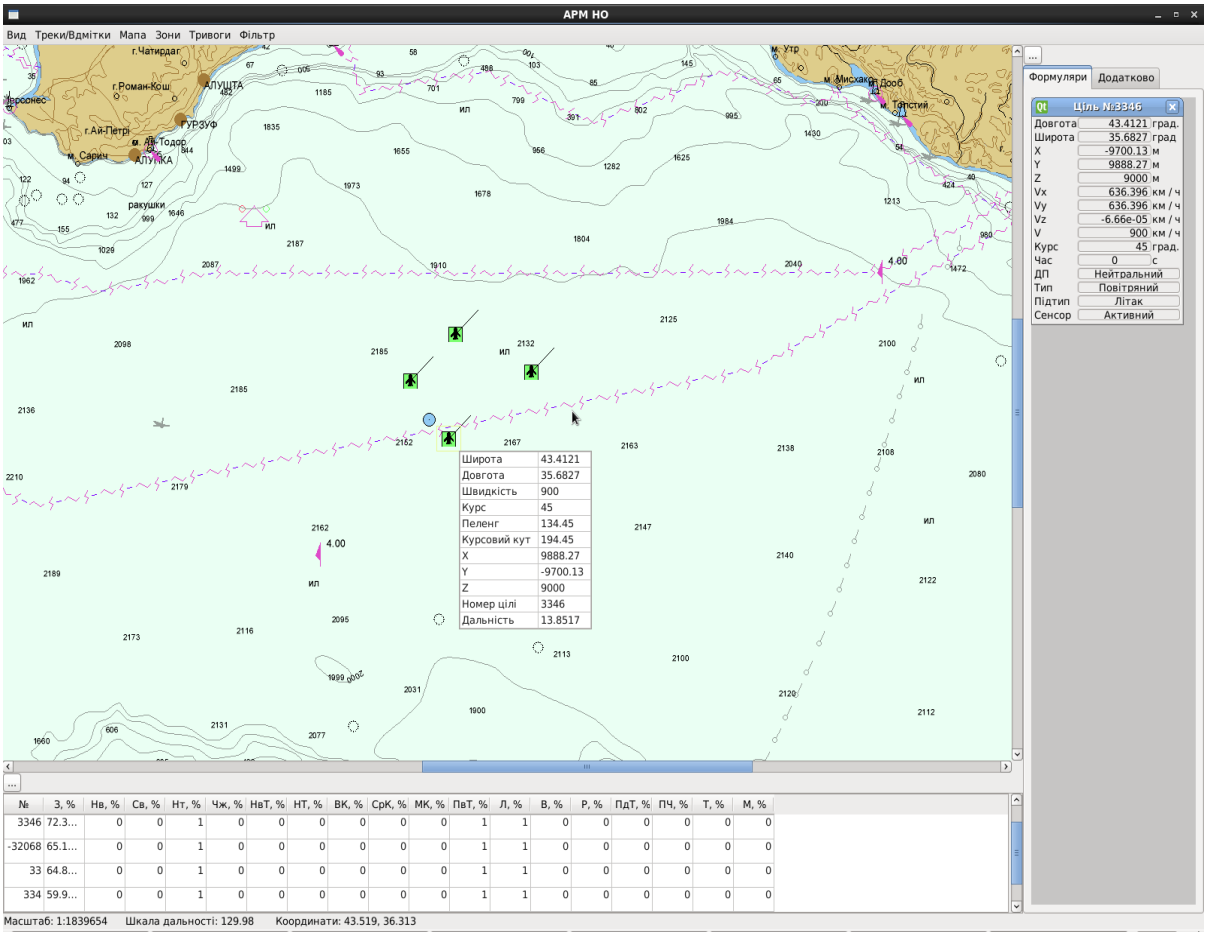

Рисунок 6.1 – Інтерфейс відображення тактичної обстановки

Інтерфейс складається з наступних частин:

1. Головне меню знаходиться у верхній лівій частині екрана і містить наступні пункти: «Вид», «Треки / відмітки», «Мапа», «Зони», «Тривога», «Фільтр».

2. Панель відображення повних формулярів цілей з кнопкою відкриття або закриття розташована в правій частині екрана.

3. Панель відображення таблиці цілей, ранжованих за загрозою, з кнопкою відкриття / закриття розташована в нижній частині екрана.

4. Кнопки зі стрілками для зміни рангів цілі.

5. Рядок стану, в якому відображаються поточний масштаб, поточна шкала дальності, географічні координати курсору та індикація тривоги.

6. Панель відображення тактичної надводної обстановки і тактичної повітряної обстановки на фоні електронної карти, яка розташована в центрі екрану.

Меню «Вид» містить наступні пункти:

– кнопка «Відкрити карту» – відкриває діалогове вікно для вибору файлу карти;

– кнопка «Вибір шарів» – відкриває діалогове вікно для вибору відображуваних шарів карти (рисунок 6.2);

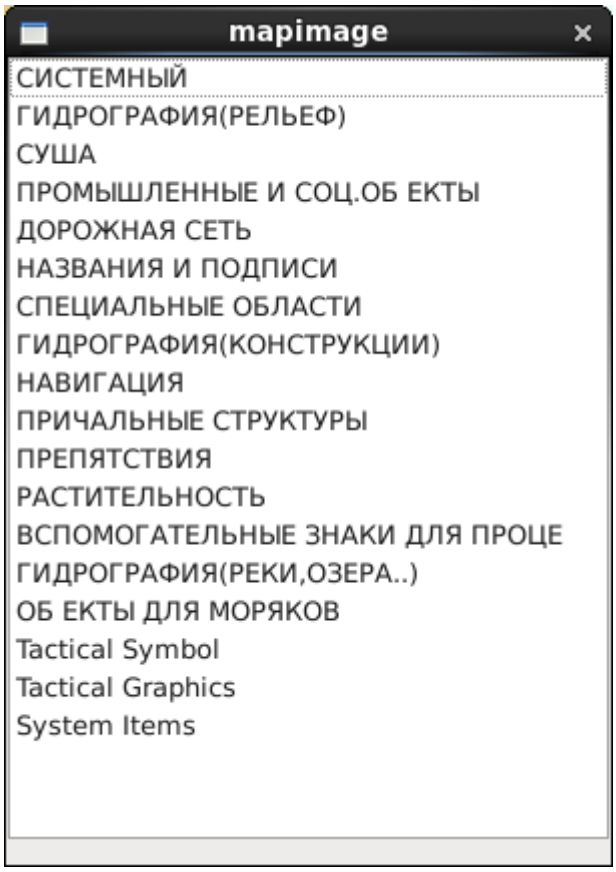

Рисунок 6.2 – Діалогове вікно відображення слоїв карти

1. меню «Режим відображення» містить пункти «Денний», «Вечірній», «Нічний» і призначене для вибору відповідного режиму відображення карти в залежності від часу доби;

2. меню «2D» містить пункти «Круговий», «Панорамний», можна вибрати відповідного виду відображення тактичної надводної обстановки і тактичної повітряної обстановки;

3. меню «Шкала дальності» містить пункти «400NM», «200NM», «100NM», «50NM», «20NM», «10NM», «Вимкнуті» і призначене для вибору відповідної шкали дальності або ж її відключення;

4. кнопка «Вихід» – закриває програму.

5. Меню «Треки / відмітки» містить наступні пункти:

– прапорець «Відображення траєкторії», який призначений для включення або відключення відображення траєкторій об'єктів;

– прапорець «Вектор швидкості» призначений для включення або відключення відображення векторів швидкості об'єктів;

– кнопка «Налаштування формуляра» відкриває діалогове вікно для вибору параметрів, що відображаються в формулярах цілей на карті (рисунок 6.3).

6. Список можливих параметрів наступний:

– «Широта» – широта цілі;

– «довгота» – довгота цілі;

– «ШВИДКІСТЬ» – швидкість цілі;

– «Курс» – курс цілі;

– «Пеленг» – пеленг цілі;

– «курсової кут» – курсовий кут цілі;

– «X» – зміщення цілі по X;

– «Y» – зміщення цілі по Y;

– «Z» – висота цілі;

– «Номер цілі» – номер цілі;

– «Дальність» – дальність до цілі.

Щоб включити формуляр для потрібної цілі, необхідно натиснути на неї лівою кнопкою мишки, щоб вимкнути – ще раз натиснути лівою кнопкою мишки на ціль. Зміна списку відображуваних параметрів в формулярі можлива тільки при відключенні всіх формулярів.

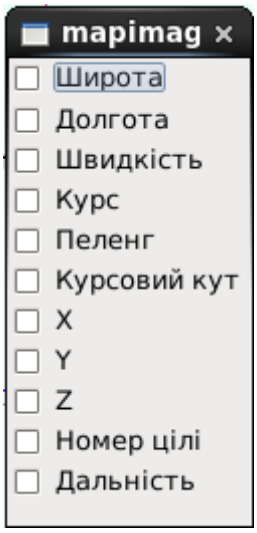

Рисунок 6.3 – Діалогове вікно для вибору параметрів, які будуть відображатись в формулярах цілей

7. Меню «Мапа» містить наступні пункти:

– меню «Масштаб», що містить пункти «1: 10000», «1: 25000», «1: 50000», «1: 75000», «1: 100000», «1: 200000», «1: 500000», «1: 1250000», призначене для вибору масштабу відображення електронної карти. Зміна масштабу також можлива за допомогою коліщатка мишки. При прокручуванні вниз масштаб збільшується, при прокручуванні вгору – зменшується;

– меню «Орієнтація», що містить пункти «Курс», «Північ», призначене для вибору орієнтації електронної карти разом з ТНО і ТПО за курсом корабля або відносно півночі відповідно.

– меню «Режим руху», що містить пункти «Істинний», «Відносній», призначене для вибору способу відображення руху нашого корабля на карті. У режимі реального руху карта стоїть на

місці, корабель рухається по карті. У режимі відносного руху корабля клрабель не рухається, карта рухається відповідно до зміни положення корабля.

8. Меню «Зони» містить наступні пункти:

9. меню «Зони огляд», що містить пункти «Фенікс-У», «Позитив-У1», «Селена», призначене для включення або відключення відображення зон видимості відповідних РЛС;

10. меню «Зони застосування зброї», що містить пункти «ОТО MELARA», «MILLENIUM», «EXOCET», «ASTER», призначене для включення або відключення відображення зон ураження відповідною зброєю.

11. Меню «Тривога» містить наступні пункти:

– прапорець «миготіння» призначений для включення або відключення візуального оповіщення про небезпечні об'єкти. Якщо він встановлений, то навколо небезпечних цілей з'являються мерехтливі рамки червоного кольору і в рядку стану з'являється мерехтлива область червоного кольору з написом «Тривога»;

– прапорець «Звуковий сигнал», який призначений для включення або відключення звукового оповіщення про небезпечні об'єкти. Якщо він встановлений, то при наявності небезпечних об'єктів видається звуковий сигнал.

12. Меню «Фільтр» містить наступні пункти:

– меню «Тип цілі», що містить пункти «Надводні», «Підводні», «Повітряні», «Невідомі» та призначене для фільтрації відображуваних цілей по типу;

– меню «Класифікація» містить меню «Держпріналежність».

– меню «Держпріналежність», що містить пункти «Свій», «Чужий», «Нейтральний», «Невідомий», призначене для фільтрації відображуваних цілей по держпринадлежності.

Панель відображення повних формулярів представлена у вигляді вертикальної області зі смугою прокрутки, в якій стовпцем відображаються формуляри обраних цілей. Щоб відкрити або закрити панель необхідно натиснути на кнопку в верхньому лівому кутку панелі. Для додавання нового формуляра необхідно натиснути правою кнопкою мишки на потрібну ціль. Для закриття формуляра необхідно натиснути на хрестик у верхньому лівому кутку формуляра.

Шляхом взаємодії з полями відображення інформації в формулярі можна коригувати дані про ціль, вводячи нові значення.

Панель відображення таблиці цілей, ранжованих за загрозою, представлена у вигляді горизонтальної області зі смугою прокрутки, в якій знаходиться таблиця. Кожен рядок відповідає конкретній цілі. Щоб відкрити або закрити панель, необхідно натиснути на кнопку в верхньому лівому кутку панелі.

Переміщення карти в області відображення можливо наступними способами:

– затиснувши ліву кнопку мишки і рухаючи курсор;

– за допомогою горизонтальної та вертикальної смуг прокрутки панелі відображення карти.

При натисканні правої клавіши миші в області карти без цілей відкривається меню, за допомогою якого до складу тактичної надводної обстановки і тактичної повітряної обстановки можна додати об'єкт або область, задати їх параметри.

ВИСНОВКИ

Комп’ютерні системи динамічного відслідковування за повітряним та надводним просторами – один з основних напрямків впровадження ІТ-технологій у системи обороноздатності України. В роботі розглянута система для кораблів типу «корвет». Основними задачами цієї системи є допомога команді корабля у прийнятті важливих рішень з приводу як власної оборони, так і оборони країни.

У роботі запропонований шар базової функціональності DDS, який задовольняє потреби системи динамічного відслідковування за повітряним та надводним просторами.

Також був реалізований алгоритм об'єднаної обробки треків цілей та алгоритми маневрування корабля. У зв'язку з цим була покращена життєздатність корабля та його команди., а також зменшений час реакції на можливі загрози.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. TACTICOS - Combat Management System // https://www.thalesgroup.com/en/tacticos-combat-management-system/.

2. LAND & NAVAL DEFENCE ELECTRONICS https://www.leonardocompany.com/en/product-services/difesa-terrestre-navale-land-naval-defence/.

3. Learn how dynamic publish-subscribe messaging can improve the flexibility and scalability of your applications. // https://www.twinoakscomputing.com/.

4. S-57 Standard Format // https://www.wartsila.com/encyclopedia/term/s-57-standard-format/.

5. SIT, StuffIt Archive (.sit) // https://hwzone.co.il/en/community/topical/15031-has-helped-convert-the-sit-format-to-any-Windows-format/.

6. Difference between C and C++ http://cs-fundamentals.com/tech-interview/c/difference-between-c-and-cpp.php/.

7. C# vs. C++: Which Language is Right for Your Software Project https://www.upwork.com/hiring/development/c-sharp-vs-c-plus-plus/.

8. Difference between java & c++ https://www.youth4work.com/Talent/Core-Java/Forum/118269-difference-between-java-c

9. Справочник штурмана под общей редакцией контр-адмирала В. Д. Шандабылова // Москва: Военное издательство министерства обороны СССР — 1968 – С. 194-271.